Universität Augsburg

Proceedings of the First Organic Computing Doctoral Dissertation Colloquium (OC-DDC'13)

Sven Tomforde (Editor)

Report 2013-06 June 2013

Institut für Informatik
D-86135 Augsburg

Universität Augsburg University

Proceedings of the First Organic Computing Doctoral Dissertation Colloquium (OC-DDC'13)

Abstract:

The Organic Computing Dissertation Colloquium (OC-DDC'13) took place in Augsburg, Germany from May 23rd to May 24th. Its aim was to bring together young researchers mainly at the begin of their PhD project to present their first ideas and share their experience in Organic Computing related research areas. The OC-DDC'13 has been organised by the Special Interest Group on Organic Computing within the Gesellschaft für Informatik (GI e.V.) and attracted 13 PhD students from all over Germany. This document contains the extended abstracts of all 13 projects and thereby gives an overview of Organic Computing-related research activities in Germany within the year 2013.

Contact

Dr.-Ing. Sven Tomforde
Lehrstuhl für Organic Computing
Eichleitnerstraße 30, R. 503
86159 Augsburg, Germany
Telefon: +49 821 598 – 4627
Email: sven.tomforde@informatik.uni-augsburg.de

Program Committee

Prof. Dr. Jörg Hähner (Universität Augsburg, Germany)
Prof. Dr.-Ing. Martin Hoffmann (Fachhochschule Bielefeld)
Prof. Dr.-Ing. Christian Müller-Schloer (Leibniz Universität Hannover, Germany)
Prof. Dr. Bernhard Sick (Universität Kassel, Germany)
Dr.-Ing. Sven Tomforde (Universität Augsburg, Germany)
Dr. Hella Seebach (Universität Augsburg, Germany)

Session Chairs

Keynote 1: Dr.-Ing. Sven Tomforde (Universität Augsburg, Germany)
Session 1: Prof. Dr. Jörg Hähner (Universität Augsburg, Germany)
Session 2: Oliver Mattes (Karlsruhe Institute of Technology, Germany)
Session 3: Christian Krupitzer (Universität Mannheim, Germany)
Session 4: Martin Jänicke (Universität Kassel, Germany)
Keynote 2: Michael Roth (Universität Augsburg, Germany)
Session 5: Sebastian Niemann (Leibniz Universität Hannover, Germany)

Contents

CASEP - Code Analysis, Speedup Estimation and Parallelization

Ioannis Zgeras

***Abstract*—To increase the performance of a program, developers have to parallelize their code due to trends in modern hardware development. Since the parallelization of source code is paired with additional programming effort, it is desirable to know if a parallelization would result in an advantage in performance before implementing it. Furthermore, parallelization of source code requires knowing about different software patterns. CASEP provides developers both with different tools to estimate potential speedup and automatically parallelization in the scope of population based algorithms.**

I. Introduction

In the last years, microprocessor development has arrived to a point where smaller integrated circuits and higher clock speeds are no longer feasible due to physical phenomena. To still increase performance, multi- and many-core architectures have become more and more important where performance gain is not only achieved by higher frequencies but primarily through parallelization. As a result, shorter execution times of application programs depend on the capability of the software engineers to parallelize their programs and thus take advantage of the parallel structures of modern computer hardware. The use of massively parallel GPGPUs (General Purpose Graphics Processing Units) as accelerators has added another level of complexity to the hardware.

One of the main challenges at achieving the best possible speedup with parallel software is to identify the best partitioning of the executable code on the hardware. This requires knowledge whether a piece of code performs better running single threaded or multi threaded on one core of the CPU, multi threaded on several cores of a multi-core CPU or multi threaded on a many-core machine such as a GPGPU. This decision is nor trivial neither automatically accomplishable. It depends on static and dynamic constraints like e.g. input size, branching factor of the code or complexity of computational operations.

Parallelizing the source code requires knowledge of different software frameworks. Some of the most known frameworks to exploit the multi-thread capabilities of multi-cores are *OpenMP* [1] or *Intel Threading Building Blocks* [2]. To exploit many-cores like GPGPUs not only knowing about frameworks like CUDA [3][4] or OpenCL [5] is necessary but also knowing about the complex memory hierarchy and cache effects. Without utilizing this complex architecture, memory access times can vary by the factor of about 10-100 clock cycles and due that have a crucial impact on the overall program performance.

As a result of all these difficulties, most developers decide to not utilize parallel hardware and implement their programs single threaded. CASEP offers the developers a tool to analyze and calculate the potential speedup of their application if they decide to parallelize it without the need of writing parallel code first. Furthermore, CASEP provides a partitioning of the source code on the given hardware to achieve the most possible speedup. Finally, CASEP is able to parallelize code segments automatically on multi- and many-core machines.

Providing reasonable speedup prediction and parallelization of general purpose software is not trivial. There are many approaches known in the literature which treat parts of the problems treated by CASEP (Sec. II). Due to the complexity of the problems most of the work issue constraints for the analyzed source code. To minimize the needed constraints the focus of CASEP is on *Population Based Algorithms* (PBAs) (Sec. II-A). Moreover, the structure of PBAs fits well on parallel hardware through the iteratively execution of independent single instructions with multiple data (SIMD) operations.

II. Related Work

An important aspect of this work deals with the identification of a best possible mapping of the Software on the existing hardware. For this purpose it must be determined which part of the code is on which hardware (CPU-mono-, CPU-multi-, many-core GPU) running fastest and/or most efficient. For the investigation of the performance of the available hardware usually benchmarks are used. As the first synthetic benchmarks are Drystone from Weicker [6] and Whetstone from Curnow and Wichmann [7] to be mentioned. Due to the growing importance of GPUs in the area of parallel computing in the last few years benchmarks to determine the performance of graphics cards have become increasingly important. While in the past the main usage of the benchmarks were mainly the determination of the graphics performance in complex 2- and 3D applications (e.g. [8]), now researchers determine also the performance of parallel computing with the help of benchmarks (see the works of M. Papadopoulos and H. Wong [9]).

A. Population Based Algorithms

PBAs are nature inspired heuristics, all PBAs have similar structures. The main part is the population that consists of a set of solutions for a given problem. These solutions are called individuals, particles or, more general, agents. These agents execute in each iteration of the algorithm different kinds of operations to improve their solution. The quality of a solution is called fitness. The function or problem that the agents

have to optimize (or solve) is called fitness function. Two representatives of PBAs are the so called Genetic Algorithms (GAs) and Particle Swarm Optimization (PSO) on which we focus in our work. More about PBAs in general and GAs/PSOs in particular can be found in [10][11][12][13].

III. CASEP Framework

This section describes the layers of the CASEP framework. On Fig. 1 an overview of the framework is shown, consisting of four layers, the investigated serial source code and given hardware as input and the parallel hardware and speedup estimation as output.

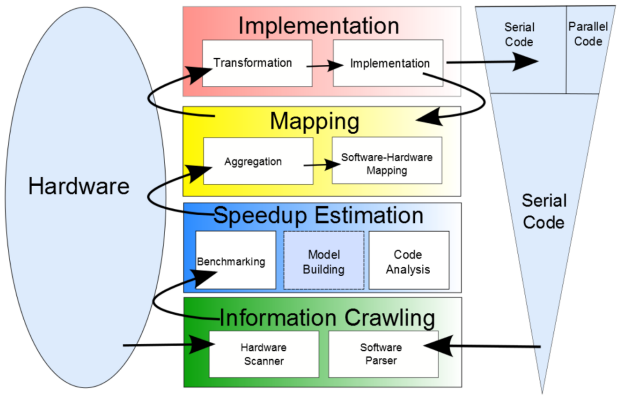

Fig. 1: CASEP Framework Overview

A. Information Crawling

The Information Crawling Layer (ICL) collects informations about the given hardware and it's specifications. These include e.g. information about the number of available cores, memory hierarchy, which is particularly relevant in GPU systems, and the data connection between computing nodes in compute clusters. Additionally, the investigated software is parsed in the ICL and transformed into an appropriate format for the overlying layers.

B. Speedup Estimation

The next layer of the framework, the Speedup Estimation Layer (SEL), is responsible for the actual calculation of the speedup. The SEL consists of three separate modules (Benchmarking, Code Analysis and Model Building) that perform the calculations in different ways.

a) Benchmarking: The benchmarking module uses synthetic benchmarks for determining the performance of the single hardware elements (CPU, GPU, etc.). The performance index is determined in terms of the possible operations that can occur in PBAs. For example, benchmarking the performance of the GPU in terms of fitness function calculation. Here, fitness functions of different complexity classes are investigated. This ensures that any fitness function, which is considered in the program, can be mapped as closely as possible by a benchmark and only a finite amount of benchmarks is necessary to represent a potentially infinite amount of fitness functions.

b) Code Analysis: The Code Analysis module uses static and dynamic code analysis to compute the possible speedup. The operations occurring in the source code and operators are each divided into logical sections (e.g. loops). For each of these sections the occurring operations are filled with test data and are executed on the existing hardware (CPU, GPU, ...) as micro-benchmarks determining their execution time. This can lead to problems whenever the runtime behavior can not be determined statically, e.g in dynamic loop iterations. In this case the runtime of the loops have to be analyzed using dynamic code analysis.

c) Model Building: The Model Building (MB) method differs from the methods described above in terms of calculating the speedup of the software mapped on parallel hardware. While the previous two methods collect informations about the machine with the help of (micro-) benchmarks, the necessary parameters of the MB method are provided by API calls or information provided by the hardware manufactures. Using this informations, two models are created, one for hardware and one for investigated software. For hardware, parameters such as number of cores, throughput, latency, memory hierarchy and pipeline stages matter while for the program mainly the kind and amount of the operations and data sizes are key parameters. These operations are used to determine the execution time on parallel hardware and thus to generate an optimal hardware/software mapping. However, this approach has some problems: First, a lot of important information are sparsely or not at all documented by the manufacturers and can not be accessed via API calls. Secondly, with this method, the compiler and hardware optimizations can only be modeled insufficient. In particular, pipeline stages and cache effects complicate the correct prediction. For this reason, MB will be used only as a coarse estimate for partitioning. The base use for this approach is the relatively low processing time which is necessary, in comparison to the other methods described above.

C. Mapping

The third layer of the framework addresses the specific mapping of software to hardware. The information collected from the Speedup Estimation layer must be connected and aggregated in a reasonable manner. Taking into account the collected informations, the single sections of the code are mapped on the dedicated hardware. That is, the hardware on which the operations can be computed as quickly as possible.

D. Parallelization

The last layer of the framework allows a partial automated parallelization of the software based on the determined software-hardware mappings by the mapping layer. However, a completely automated parallelization is not feasible. For this reason, some criteria for the parallelization will need to be set, such as the restriction to population-based algorithms and the programming templates and directives.

Environments in which population-based algorithms are used are typically dynamic. The environment may change during the execution of the algorithm and thus modify the fitness

function, but also other parameters of the algorithm. Therefore, the computed software-hardware mapping on compile time has to be modified. This circumstance faces the CASEP framework with a constant recalculation of the speedup and prediction of the mapping at run time. To not influence the execution time of the program the overhead of the CASEP framework has to be determined and single modules have to be disabled, if necessary, or at least modified.

IV. Evaluation

The main tasks of CASEP frameworks are the prediction of the potential speedup and parallelization of individual sections. As a result of the tasks the basic metrics for the evaluation are: The speedup calculated must be compared with the actual speedup that results when the serial code is parallelized. In addition, the quality of the automatically generated code has to be compared to both the calculated speedup, and with a manually generated optimal parallel version. It is furthermore important that the results of the parallelized version match that of the serial program.

To compare the values a reference implementation of a PSO algorithm should be implemented. This reference implementation should achieve an optimal runtime on HPC computing cluster and on desktop computers using parallel implementations from the literature and optimization tools. To optimize the reference implementation Nsight, a tool provided by NVIDIA, should be primarily used that provides interfaces for memory management, bandwidth measuring and CUDA debugging functionalities. If for the evaluation a second algorithm is needed an optimized version of a GA should be used, similar optimized as the PSO described above.

To use the analysis provided by CASEP at runtime the overhead has to be investigated. In particular, the overhead produced may not exceed the speedup achieved by the parallelization.

V. First Results and Future Work

The different modules of the SEL have already been implemented and evaluated. In our first evaluation scenarios the Benchmarking [14] and Code Analysis [15] modules achieve reasonable evaluation results. However, for both approaches different optimizations have to be implemented. The Benchmarking module needs informations about the source code that are provided by the programmer. The evaluation results have shown that the current informations are not sufficient to model the benchmarks needed to compute the speedup. In particular, the abstraction using complexity classes can be insufficient and other characterization criteria have to be investigated. The first approach of the Code Analysis module used only static code analysis. The evaluations showed that the correctness of the speedup prediction using static code analysis only depends mainly on the knowing of the iteration count. If this information is known at compile time, the prediction is accurate, otherwise the iteration count has to be predicted and the speedup prediction is inaccurate. First results with dynamic code analysis to face this problem have shown that dynamic code analysis can be used to minimize this inaccuracy. Preliminary evaluation results show that the MB module computes the speedup of source code accurate if the functions contain mainly memory operations. The more compute operations exists the more the computed speedup differs from the real speedup. These results show that the model has to be reworked to map the compute operations more precisely.

The last step is to merge the different speedup estimation modules and aggregate their results to achieve more precise results. These aggregated speedup will be used for the parallelization module to parallelize the single source code segments.

References

[1] L. Dagum and R. Menon, "Openmp: an industry standard api for shared-memory programming," *IEEE Computational Science and Engineering*, vol. 5, no. 1, pp. 46–55, 1998.

[2] C. Pheatt, "Intel threading building blocks," *J. Comput. Sci. Coll.*, vol. 23, no. 4, pp. 298–298, Apr. 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1352079.1352134

[3] NVIDIA, *NVIDIA CUDA Compute Unified Device Architecture - Programming Guide*, 2007.

[4] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym, "Nvidia tesla: A unified graphics and computing architecture," in *IEEE Micro*, vol. 28, no. 2. Los Alamitos, CA, USA: IEEE Computer Society Press, 2008, pp. 39–55.

[5] J. E. Stone, D. Gohara, and G. Shi, "Opencl: A parallel programming standard for heterogeneous computing systems," *Computing in Science and Engineering*, vol. 12, pp. 66–73, 2010.

[6] R. P. Weicker, "Dhrystone: a synthetic systems programming benchmark," *Commun. ACM*, vol. 27, no. 10, pp. 1013–1030, Oct. 1984. [Online]. Available: http://doi.acm.org/10.1145/358274.358283

[7] H. J. Curnow, B. A. Wichmann, and T. Si, "A synthetic benchmark," *The Computer Journal*, vol. 19, pp. 43–49, 1976.

[8] "3dmark." [Online]. Available: http://www.3dmark.com/de/

[9] H. Wong, M. myrto Papadopoulou, M. Sadooghi-alv, and A. Moshovos, "Demystifying gpu microarchitecture through microbenchmarking," in *In ISPASS*, 2010, pp. 235–246.

[10] M. Mitchell, *An Introduction to Genetic Algorithms (Complex Adaptive Systems)*, third printing ed. A Bradford Book, Feb. 1998. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0262631857

[11] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine Learning*, vol. 3, pp. 95–99, 1988, 10.1023/A:1022602019183. [Online]. Available: http://dx.doi.org/10.1023/A:1022602019183

[12] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, nov/dec 1995, pp. 1942 –1948 vol.4.

[13] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intelligence*, vol. 1, pp. 33–57, 2007, 10.1007/s11721-007-0002-0. [Online]. Available: http://dx.doi.org/10.1007/s11721-007-0002-0

[14] I. Zgeras, J. Brehm, and M. Akselrod, "Function based benchmarks to abstract parallel hardware and predict efficient code partitioning," in *Architecture of Computing Systems (ARCS), Proceedings of 2013 26th International Conference on*, 2013, pp. 1–13.

[15] T. S. Ioannis Zgeras, Jürgen Brehm, "A model based approach for computing speedup on parallel machines using static code analysis," in *Parallel and Distributed Computing and Systems*, 2011.

An Autonomous Self-Optimizing Memory System for Upcoming Manycore Architectures

Oliver Mattes
Institute of Computer Science & Engineering (ITEC)
Karlsruhe Institute of Technology (KIT)

***Abstract*—With the rising number of CPU cores manycore processor systems will be the upcoming future system structure. Today's manycore processors like Intel SCC, Tilera TILE or KALRAY's MPPA are primarily designed for high performance applications, using several cores with direct inter-core communication and avoiding external memory accesses. Up to now the system memory offers access over a limited number of controllers, which leads to congestion or inefficient memory assignment.**

But the spreading of manycore systems in the near future brings up application scenarios with multiple concurrently running dynamic applications, changing I/O characteristics and a not predictable memory usage. Hence, dynamic allocation of memory, especially of shared memory, is necessary, and the memory management must become more flexible and distributed in nature. Further, integrated self-organization mechanisms enable transparent optimization of physical memory resource utilization e.g. for latency or energy efficiency issues.

Our evaluation results show that a decentral management is feasible and the optimization approach for the memory assignment can be employed with only a small overhead of additional messages over the on-chip network.

I. Introduction

The continuously increasing integration level of CMOS devices is leading to manycore systems. As yet these systems are designed for high performance applications running on several cores using direct inter-core communication over shared on-chip caches or small per-core memories, trying to avoid external memory accesses.

So far the system memory commonly consists of one or a few memory components and offer access to memory over a limited number of controllers. The difference between the uprising core count and the slow external memory, the so called memory wall [1], is also getting more important with the increasing number of cores. This lack in the memory system leads to congestion [2] or inefficient memory assignment, getting worse with an increasing number of heterogeneous cores. Moreover with multiple programs running concurrently, memory must be managed dynamically, and because of data locality, placement restrictions and memory regions already occupied by other tasks, an optimal dynamic assignment of memory regions to tasks is not possible in most cases.

In order to scale the memory with the rising core count and to tackle the problem of optimizing the management and assignment of memory to tasks in dynamic scenarios, we propose Self-aware Memory (SaM) [3]. In this extended abstract we present this scalable memory management system for adaptive computing systems and an optimization mechanism for systems without a central decisive instance, which is used for continuous verification and optimization of the system. To achieve that, we have to ensure some challenges: high flexibility for reacting on the high dynamic application scenario, scalability of the memory management for handling the upcoming increase in cores and concurrent tasks, and a continuous self-optimization to adapt and optimize the system state for the actual running applications. These challenges and their realization are presented in the following chapters.

II. Related Work

In the last years first manycore systems came up in research and industry, but so far only a few of them are commercially available.

The Intel SCC (Scientific Cloud Computer) is a project to evaluate different challenges of future manycore architectures, e.g. the network-on-chip, the tiled architecture, communication structure and programming models. The main purpose is executing message passing applications which communicate over a distinct per-core and cache-like memory. The external memory is accessed over four memory controllers and the initial memory assignment has to be done manually in advance of executing applications. The latency for accessing the external memory modules also differs strongly depending on the used compute core [4].

The Tilera TILE-Gx processors are a group of commercially available manycore processors and a follow-up of the MIT RAW project [5]. As well as the Intel SCC the Tilera systems use a tiled architecture and are designed to execute parallel applications like streaming applications, which mainly communicate directly between the cores, therefore using a shared cache [6]. Access to the external memory, depending on the core count of the processor, is achieved over one to four memory controllers.

KALRAY's MPPA (Multi-Purpose Processor Array) manycore [7] is designed mainly as a dataflow architecture. The VLIW cores of the processor are grouped to clusters, containing a system core and integrated memory. The MPPA could be programmed by a c-based parallel dataflow model or with POSIX C/C++, which enables threading within a single cluster. The external main memory is also connected over two memory controllers.

Within a DFG-funded research project the Digital On-Demand Computing Organism for Real-Time Systems (DodOrg) [8] was examined. DodOrg is computer architecture

which is based on OC principles to explicitly exploit self-adaptation processes. Such processes are especially used for internal communication, energy and power consumption management, and steering application processing. For this, suitable mechanisms found in biological organisms are adopted such as e.g. the internal communication, which is inspired by the biological hormon system.

The integration of self-x functionality in systems, also known as organic or autonomic computing, is a research area, which was tackled since the last decade. A visionary overview was given in [9] in which the structure of autonomic elements was described as basic principle of self-managing systems. These autonomic elements consist of the managed element and an corresponding autonomic manager. This manager runs the so called MAPE cycle, to monitor, analyze, plan and execute the management task based on informations by underlying self-knowledge. This principle is captured by the proposed decentral memory system and the self-optimization process.

In [10] associative counter arrays are introduced to accumulate and pre-process monitoring informations. In case of an overflow of a counter a status message is sent to the next upper monitoring instance in the hierarchy, up to a central instance which processes the collected informations and initiates a reaction. Instead of an centralized management instance, in the presented work, the monitoring and the subsequent optimization process is handled by a cooperation of the self-managing components of a decentralized system. Therefore new ways of finding optimization possibilities in decentralized calculating of optimization advices and a consensus building process have to figured out.

III. Challenges

A scalable and adaptive memory management system for manycore systems has to handle some challenges which are addressed in the following. In addition the enhancements for a continuous self-optimization are presented.

A. Flexibility

With a high dynamic application scenario an initial or pre-configured assignment of memory resources is no longer feasible. Dynamic memory management is needed to fulfill different memory requirements and to offer the ability to use varying memory hierarchies and speeds. With a higher flexibility memory blocks on all available memory locations are possible and a restrictive statical assignment of memory to a group of cores is no longer necessary.

We propose Self-aware Memory (SaM) in which the memory is partitioned into several self-managing component to achieve this flexibility. As seen in Figure 1 on CPU core and memory side, SaM components are added to organize the distributed memory management. Memory allocation during run-time is done with client-server principles building an abstraction layer so that applications running on the cores don't have to be modified. The SaM components transparently realize virtual to physical addresses translation as well as Transactional Memory (TM) like synchronization mechanisms

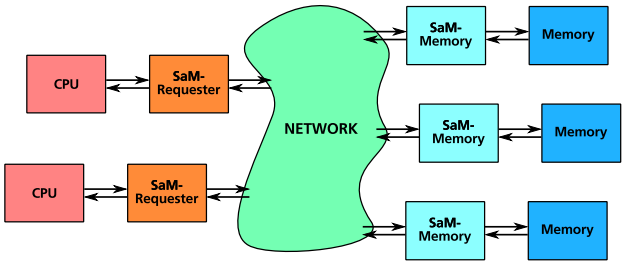

Fig. 1. Distributed SaM structure with assigned management components

for access to shared memory regions. Memory coherency is organized by the memory system using TM principles with a combined HW and SW mechanism [11]. This provides the programmer an easy way for accessing shared memory and an abstract view of the memory resource.

B. Scalability

With the ongoing increase of compute cores, the number of memory components and controllers has to be increased as well, otherwise the few modules get contented at simultaneous accesses. To avoid congestion or outages, collecting all informations about the system in a single central instances should be avoided. With a proposed decentral system structure as of SaM the memory components are responsible exclusively for their own, and accesses can be better balanced. Outages are isolated and the dynamic management could reorganize the memory system.

In more detail with SaM the memory is divided into several autonomous self-managing memory modules, each consisting of a management component and a part of the physical memory. The individual memory modules act as independent units and are no longer directly assigned to a specific processor. The memory allocation is achieved by a cooperation of the available autonomous components. Self-awareness means, that these autonomous memory components have knowledge about the assigned memory blocks, their access rate, ownership and access rights. And in addition they are aware of their own state e.g. physical condition using CRC checks and the state of their neighborhood. The basic SaM mechanism is independent of a distinct network structure or hierarchy. However the mechanism could be adjusted and fine-tuned depending on the actual network. Several structures as grids, stars or buses, in different hierarchies are possible.

C. Continuous Self-Optimization

Executing high dynamic application scenarios with multiple running concurrently applications, an optimal dynamic assignment of memory regions to tasks is not possible in most cases because of data locality, placement restrictions or already occupied memory. Dynamic changing application needs and memory requirements necessitate an on-the-fly optimization. Moreover hand-optimization based on prior knowledge is not practicable due to the high dynamic behavior. Further usage of heterogeneous cores strengthens the allocation problems

because the applications are pinned to distinc cores so the memory has to be migrated.

To fulfill these needs we combined the decentral and adaptive SaM memory system with self-optimization mechanisms which is presented in a more detailed view in section IV.

IV. Decentralized Self-Optimization

A. Optimization Process

The proposed optimization process is based on the MAPE cycle (Monitor, Analyze, Plan and Execute) and uses the following steps:

1) Decentralized Monitoring and Data Preprocessing: local data collection per system component and periodic exchange with neighbors.
2) Data Analysis: analysis of the monitored information, including associative counters, which provide a threshold value for the following optimization step.
3) Optimization Algorithms: initiated by the overflow of an associative counter, in this step an optimization advice is calculated using a dedicated optimization algorithm.
4) Decentralized Consensus Building: Validation of the proposed optimization advice and decentralized voting procedure.
5) Optimization: The actual execution of the accepted advice. Depending on the optimization algorithm this might be a data migration process combined with an update of the address management tables. The virtual memory addresses on CPU side are not modified.

After this steps an optimized system behavior is achieved for the moment. This is an coincident optimization process, ongoing on all system components to react on dynamic changes.

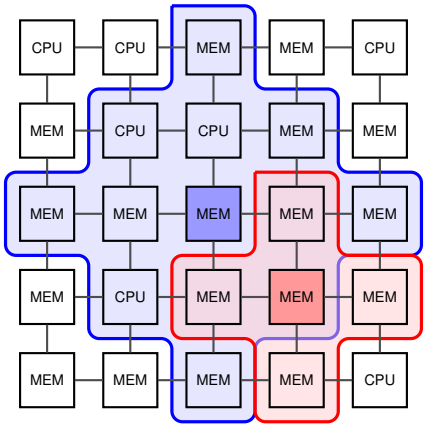

Fig. 2. Distribution of informations with different radius sizes

B. Decentralized Monitoring

As the first step in the optimization process the independent system components have to collect informations about themselves and their neighborhood, e.g. the status changes and its memory usage.

First of all, each component has to collect informations about itself. Then in periodical intervals informations about the condition of the components has to be exchanged between neighbors. On typical NoCs this has to be realized using explicit messages. This could be done by broadcasts of a restricted length. Depending on this maximal number of hops the size of the neighborhood differs. Using more hops leads to a more global system knowledge for global optimization, but results in a higher overhead by the numerous monitoring messages. Lower hops restrict the number of monitoring messages on the network, but the optimization only could be done on a bounded local region. A trade-off between these two principles has to be done. Two examples of broadcast with a radius of one (red) and two (blue) are provided in Figure 2.

By exchanging informations with the neighbors, the standalone components cumulate informations. Integrating these additional informations in their own condition messages the information and the view of the system grows step-by-step on each distributed component. As an example the distances and bandwidth of connections between the components of the network could be figured out and updated in case of dynamic changes.

C. Distributed decision making process

In the underlying loosely coupled system structure, an optimization cannot be initiated by a central instance. The optimization has to be accepted by all self-managing participants in a decentralized consensus building process. Discovering an optimization potential, a memory component makes up an optimization advice, which is then provided to all participants. Each participant decides if the received advice makes sense for itself and sends a confirmation or rejection. The decision of each participant is based on its own collected information basis, so with n participants n decisions are done. If a participant agrees to an advice, it sets a flags and doesn't participate on other optimizations.

The exchange of advice and answers for the decentralized consensus building could be done in several ways. We implemented and evaluated the three message exchange methods: three-way handshaking, ring-mode and broadcast. All three methods have advantages and disadvantages. They differ in the number of messages for the consensus building. The number of messages with the ring method and three-way handshaking grows linear, at broadcast it grows quadratic.

D. Global vs. Local Optimization

The presented optimization process is aligned to several concurrent local optimizations by the different self-managing components of the system. This approach is theoretically justified by the decentralized decision making for multi-agent systems [12], describing decision making with several instances, called agents, and negotiations without any central instance. Any single i of the n agents makes his own decision, represented by the decision vector $v_i \in V_i$ which are combined in the global decision vector

$$v = \left[v_1^T, v_2^T, ..., v_n^T\right]^T$$

The global decision vector contains the latest decision vectors of the individual agents. The optimal global decision v^* then is calculated as

$$v^* = arg \min_{v \in V} J(v)$$

$$\text{with } J : V_1 \times V_2 \times \cdots \times V_n \rightarrow \mathbb{R}$$

as objective function.

Concerning the ongoing refinement of the multiple vectors, the decision information from the system components could be outdated, which is why it is often not possible to find a global optimal decision. Regarding the high dynamic workload this problem is enhanced, because all system changes result in updated decision vectors and a stable global decision vector is not achieved. A system-wide information distribution also doesn't scale with rising system size, because the number of necessary messages is getting too big. Therefore a global optimization is not reasonable for the assignment. In the underlying scenario, several applications are also locally bound to a distinct part of the system, in which then a local optimization could be done.

V. Evaluation

We developed different prototypes for evaluating the decentral memory system and the self-optimization process. First of all a SystemC-based SW simulation is used to evaluate protocol enhancements and to classify the several possibilities in the optimization process, monitoring and decentral decision making. We further on created a FPGA-based HW prototype and a SW daemon, enabling to run a SaM service on normal Linux systems. Currently we are starting to port this SaM service to a system with a Tilera TILE-Gx processor, enabling dynamic memory allocation and run-time optimization.

Eval	Optimization step
20	Information exchange
3	Consensus building
1	Data migration
3	Actualization of management tables
27	Total

TABLE I
NUMBER OF MESSAGES AT MINIMAL COMMUNICATION OVERHEAD FOR ONE OPTIMIZATION

Evaluations using the SystemC simulation with different system and memory structures using traced benchmarks, showed that a decentral memory management is feasible and that the optimization approach for the memory assignment can be employed with only a small overhead of additional messages over the on-chip network. Table I shows the result of an evaluations which confirms out theoretical calculations and predictions. It shows the numbers of additional messages for this single optimization process from the beginning of the application up until the first data locality optimization. These additional messages are the difference between an optimized execution and running the same scenario without the optimization mechanism.

VI. Conclusion and Outlook

This extended abstract presents a decentralized autonomous memory architecture with self-optimization capabilities. The presented mechanisms enable the continuous verification and optimization of the management and assignment of memory to already running applications in a system without a central decisive instance. Evaluations showed that with a small number of interchanged state messages, an optimization could be initiated and executed. Depending on the collected information basis, several concurrent local optimizations could be performed, resulting in a better performance of the system. Regarding the assumed high dynamic application scenario only parts of the system are used for one particular application. Along with the rapidly outdated decision information and the not scalable amount of monitoring information in a central instance, multiple local optimizations are favorable to global ones. Potential and temporary disadvantages for individual applications are going to be detected and re-optimized by the persistent optimization process.

Adapting and evaluating the mechanism for new and upcoming memory connections like 3D-stacked memory associated with the changing system structure is another step on our agenda.

References

[1] W. A. Wulf and S. A. McKee, "Hitting the memory wall: implications of the obvious," *SIGARCH Comput. Archit. News*, vol. 23, no. 1, pp. 20–24, Mar. 1995. [Online]. Available: http://doi.acm.org/10.1145/216585.216588

[2] J. Duato, "Beyond the power and memory walls: The role of hypertransport in future system architectures," in *First International Workshop on HyperTransport Research and Applications (WHTRA)*, February 2009.

[3] R. Buchty, O. Mattes, and W. Karl, "Self-aware memory: Managing distributed memory in an autonomous multi-master environment," in *Architecture of Computing Systems – ARCS 2008*, ser. Lecture Notes in Computer Science, vol. 4934, February 2008, pp. 98–113.

[4] G. W. Intel Labs China, "Scc architecture and design overview," Intel Labs Single-chip Cloud Computer Symposium, December 2010.

[5] M. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, J.-W. Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal, "The raw microprocessor: a computational fabric for software circuits and general-purpose programs," *Micro, IEEE*, vol. 22, no. 2, pp. 25–35, 2002.

[6] T. Corporation, "Tile-gx processor specification brief," May 2012.

[7] Kalray, "Kalrays mppa (multi-purpose processor array)," http://www.kalray.eu/products/mppa-manycore/, April 2013.

[8] J. Becker, K. Brändle, U. Brinkschulte, J. Henkel, W. Karl, T. Köster, M. Wenz, and H. Wörn, "Digital on-demand computing organism for real-time systems," in *ARCS Workshops*, 2006, pp. 230–245.

[9] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, pp. 41–50, January 2003.

[10] D. Kramer, R. Buchty, and W. Karl, "Monitoring and self-awareness for heterogeneous, adaptive computing systems," in *Organic Computing - A Paradigm Shift for Complex Systems*, ser. Autonomic Systems, C. Mller-Schloer, H. Schmeck, and T. Ungerer, Eds. Springer Basel, 2011, vol. 1, pp. 163–177.

[11] O. Mattes, M. Schindewolf, R. Sedler, R. Buchty, and W. Karl, "Efficient synchronization techniques in a decentralized memory management system enabling shared memory," ser. PARS Mitteilungen GI, vol. 28, no. ISSN 0177-0454. Rüschlikon, Switzerland: Gesellschaft für Informatik e.V., May 2011.

[12] G. Mathews, H. Durrant-Whyte, and M. Prokopenko, "Decentralized decision making for multiagent systems," in *Advances in Applied Self-organizing Systems*. Springer London, 2007, pp. 77–104.

A Distributed Controller for Organic Computing Applications

Stefan Rudolph
Organic Computing Group
University of Augsburg
stefan.rudolph@informatik.uni-augsburg.de

***Abstract*—In the field of Organic Computing, there is a trend to shift the adaption of systems to its application from design-time to run-time. This makes the presence of some kind of learning component indispensable. The basic assumption for this work is that in systems with multiple entities, the knowledge of dependencies in this system can give a huge advantage in terms of a faster and better learning of the best behavior in many situations. In this work, I propose a working plan for my PhD studies that outlines an approach to solve this issue. The planned work includes the development of a formal framework for the modelling of dependencies, techniques for the automated detection of these dependencies and a distributed controller which integrates this techniques.**

I. Introduction

In the field of complex systems, there is a huge expenditure of time for the developer in order to identify all possibly occurring events and find appropriate reactions to this events. Organic Computing [1] tries to solve this issue by shifting the adaption of systems to its application from design-time to run-time. To reach the goal of an adaption of the system to new situations without the intervention of humans, some kind of learning component has to be integrated in the system. The learning mechanisms available today only implicitly take into account the dependencies between entities in the system. This work presents an approach to make this dependencies an explicit factor in learning techniques in order to fasten the learning of better solutions.

The rest of the paper is structured as follows: In section I-A, the Observer/Controller (O/C) architecture in which the new techniques will be integrated is presented in short. In section I-B, the related work for the modelling and calculation of dependencies is described. In section II, a working plan, including the major issues and approaches to solve them, is outlined.

A. The Observer/Controller Architecture

This O/C architecture has been initially proposed in [2], therefore only a brief discussion follows. The overall goal of the architecture is to give a generic framework as a blueprint for future Cyber-Physical-Systems (CPS), although the architecture has a more general character and can also be applied to other settings besides CPS. This three-layer node architecture is depicted in Fig. 1. A major novel aspect of this architecture is to enable a strong collaboration between individual nodes in the system, in order to increase the speed at which the system as a whole can adapt itself to new situations. The proposed architecture separates different (environmental) scopes of action in order to implement the most suitable methods for each scope. At the first level, the *reaction layer*, local adjustments are made on a single node of the system ensuring fast and robust reactions to recurring, known situations. The second level, the *cognition layer*, extends the system's scope of action on a single node by longer-term observations and the application of machine learning techniques in order to extend the capabilities of the reaction layer towards unconsidered situations and anomalies in the environment. The third level, the *social layer*, enables the cooperation between multiple nodes of the system in order to react to situations that arise in a broader scope than what a single node can address. This includes joint actions of multiple nodes that require coordination in order to be successful.

Besides other details of the architecture, in the context of this work, the assumption that the reaction layer is rule-based is especially interesting. Each rule basically has the form:

$$[CONDITION][ACTION][RATING]$$

In this context, *CONDITION* is a situation model which allows for a determination of the degree of applicability each rule has for a given environmental situation. *ACTION* defines the output of the system that follows if a current input "matches" the given situation. Finally, *RATING* summarizes properties such as "usefulness" of a certain rule in the past, which are used to weight certain rules in certain situations.

Within this architecture, my work will focus on the development of techniques for the controller on the social and cognition layer.

B. Related Work

A starting point for the dependency modelling of parameters can be found in mathematics and Operation Research. In mathematical finance, the idea is to quantify dependencies between random variables, which are interesting for the modelling of credit default risks. A modern approach to this is the use of copulas. Copulas can model dependencies between random variables much more accurately than other measures and give them a huge advantage over simple techniques such as the covariance [3]. Another related approach are risk measures [4] that are heavily used in the field of Operation Research in order to quantify risk in the economic world. Commonly used

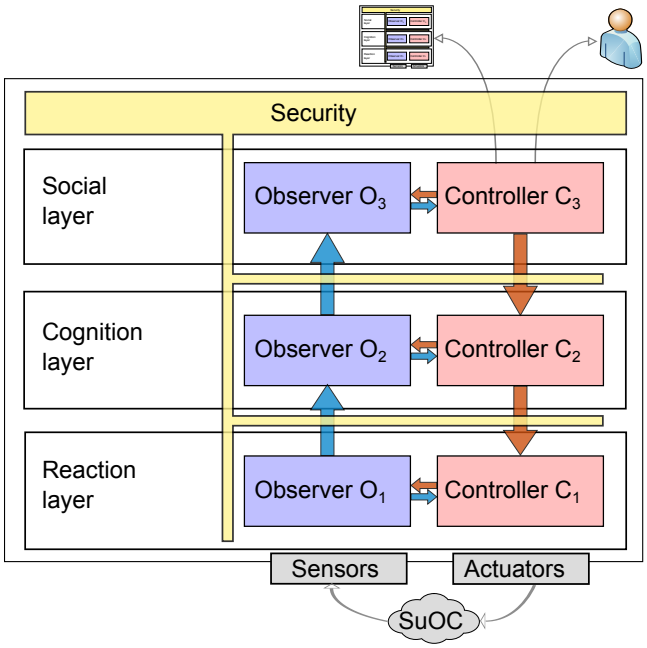

Fig. 1. Three-Layered O/C architecture

is, e.g., the Value-at-Risk approach. In this context, the risk is a value that is aggregated from the probability of a loss and the size of loss. Most of the time, it is necessary to estimate the dependencies between the random variables in order to calculate the risk. The afore described techniques are useful for the detection of dependencies. In order to determine the actual dependencies, including a possibility to accelerate the learning process of the system, there are several interesting approaches that might be adapted such as regression analysis [5], factor analysis [6], or principle component analysis [7].

II. Working Plan

In my PhD studies, I will investigate four major topics that are described in this section. First, I will develop a taxonomy regarding the *ACTION* spaces in OC systems (see section II-A). Second, I will investigate techniques for the modelling of dependencies and methods for the automated retrieval of such dependencies without the restrictions, such as, network bandwidth (see section II-B). The next step is to adapt this techniques to OC systems, such as, systems with local information only (see section II-C). The last step is to integrate the developed framework and techniques in a controller architecture (see section II-D).

A. Development of a taxonomy

Here, the goal is to develop a taxonomy for the *ACTION* spaces in OC systems. The assumption is that recurring patterns in the properties of these applications occur that allow researchers and developers to find generic O/C architectures that enable a proper functionality in the addressed application classes and thereby can be easily reused.

As a starting point for the classification, the large collection of Organic Computing applications that is examined in detail in [1] can be used. There are especially two taxonomies that use a very elaborated methodology: The first is a taxonomy for complex systems [8] and the second is for mobile applications [9] that have to be considered. Both of them use a method that combines the empirical-to-deductive and the deductive-to-empirical approach in order to find an adequate amount of characteristics and classes that are important. There are other classifications, for instance, regarding uncertainty [10], [11] that might be useful as inspiration for such characteristics.

B. Development of modelling approaches for dependencies in the ACTION space of intra-node learning systems

The goal of this work package is to find suitable approaches for modelling dependencies in the *ACTION* space of CPS. In this WP, the focus is on *local* dependencies, i.e., dependencies between atomic *ACTION*s, where no communication overhead is needed to detect these dependencies. Inter-node dependencies are an issue in section II-C.

These objectives have to be seen in the context of the afore introduced 3-layered architecture of the node controller. Within this architecture, the function of the controller in the cognition layer is to create new rules that are applied and evaluated in the reaction layer. These rules will exploit the dependency structures in the application. The task of the controller in the social layer is to gather information about the inter-node dependencies of the system and to forward relevant information to C_2 in order to further improve the rule creation process.

The approach starts with an abstract mathematical description of the requirements for the dependency measures within the *ACTION* space that will serve as a basis for the more concrete modelling of these dependencies. In all applications, the *ACTION* space can be described as a space $K = V_1 \times V_2 \times \ldots \times V_n$ where the V_i are the *ACTION* spaces of a single parameter. This means, the whole *ACTION* space is just a combination of the n elementary subspaces of the *ACTION* space. The objective is here to find a function $\alpha : \mathcal{P}(\{1,\ldots,n\}) \times \mathcal{P}(\{1,\ldots,n\}) \longrightarrow [0,1]$, which quantifies the degree of the dependency between two arbitrary subspaces of the *ACTION* space in a range from 0 (no dependency at all) to 1 (strongest possible dependency). It is possible to formulate abstract requirements for this function α, for example

Self-Independence: The function α fulfills the property of self-independence if $\alpha(P,P) = 0 \quad \forall\ P \subseteq \{1,\ldots,n\}$. This property is natural because every sub$ACTION$ space should be completely independent to itself.

Symmetry: The function α is symmetric, if $\alpha(P_1,P_2) = \alpha(P_2,P_1) \ \ \forall\ P_1,P_2 \subseteq \{1,\ldots,n\}$. This property implies that all dependencies are in effect in both directions.

These are example properties that have to be extended, but are valuable for the development of dependency measures for specific systems.

A starting point for the more concrete modelling within this framework can be found in mathematics and Operation Re-

search. There are also promising approaches in mathematical finance. The concept of dependency measures (in stochastics, cf. section I-B) is especially interesting in systems with only local information. A possibility is to interpret the local fitness and the atomic *ACTION*s as random variables and quantify the dependencies between them. The usefulness of this technique in the machine learning domain will be analyzed. Another approach are risk measures (see section I-B). In this context, the risk is a value that is aggregated from the probability of a loss and the size of loss. This notion of risk can be interpreted as a measure of dependencies in the way that atomic *ACTION*s have a high dependency if and only if there is a high risk. The application of such risk measures to the field of machine learning will be analyzed.

In order to exploit the knowledge of the dependencies, it is not only necessary to quantify them by using the presented techniques but to describe the actual manifestation of the dependencies. Suitable representations for the dependencies could be functions, inequations and propositional or temporal logic. For the creation of such representations, different tools may be used. For example the mathematical techniques regression, principle component analysis or factor analysis can be adapted to the problem or Genetic Algorithms might be used.

C. Development of modelling approaches for dependencies in the ACTION space of distributed learning systems

The goal of this work package is to extend the techniques developed in section II-B for the detection and exploitation of intra-node dependencies for inter-node dependencies. In the preceding work the focus was on intra-node dependencies in the *ACTION* space. This ignores some issues that occur in a distributed OC system, e.g., the negative effects of a huge communication overhead and problems related to a heterogeneous network of nodes, such as different learning strategies and conflicting goals.

Here, the following research issues will be considered. (1) Looking at the inter-node dependencies in the *ACTION* space, it is crucial to find techniques to smartly limit data exchange for the detection of such dependencies. The objective is to find and analyze approaches to keep the communication overhead as low as possible. On the one hand, this can be reached through data selection, i.e., the relevant data will be spread but not unimportant (or less important) data, or data aggregation, for example, by merging multiple values over time to a single one. With these data it will not be possible to analyze the dependencies as well as with the raw data, but it is sufficient to identify the important nodes. Knowing these nodes, the communication with them can be increased and the detailed dependency structure can be discovered. (2) Considering heterogeneous systems, it might be applicable to use different learning techniques and strategies in different nodes. In this case it is necessary to integrate the knowledge about dependencies in all the nodes, i.e., in all the different learning strategies used at the cognition layer. For this purpose I will develop a generic representation of the dependencies based on the ones developed in section II-B and integrate them into different learning techniques.

Hence, here, the focus is on the cooperation between nodes by retrieving information about dependencies and exploiting this knowledge in a distributed system. This will provide the applicability in (heterogeneous) distributed OC systems.

D. Development of distributed controller architectures

The goal of this work package is to develop suitable architectures for the interaction between the nodes. Of course, it is very likely that the optimal architecture differs from one application to another, but general conclusions regarding the general classes of applications should be possible.

There are three basic architecture approaches to consider: P2P systems [12]–[14], hierarchical systems [15] and hybrid systems [16].

The approaches will be analyzed for the use in a distributed controller regarding the applicability to different classes of CPS. An assumption is that it is possible to find patterns in the utility of architectures in different classes of applications.

Besides the basic structure of the architecture of the system, there is another research issue that is the communication paradigm. There are several options for this purpose in general. First, it is possible to use a message-based communication system in which the node communicate by sending messages to each other. Second, a state-based approach is investigated in which the nodes are able to ask for the state of a second node in order to decide appropriately. Furthermore, publish/subscribe approaches will be investigated.

III. Conclusion

The main contributions of this work will be the development of the formal framework for the modelling of dependencies and the investigation of techniques for the automated detection of this dependencies, as well as, the development of a distributed controller and the integration in the presented generic O/C architecture. In order to test the applicability of the developed techniques, the principles will be applied to at least two scenarios, a self-organizing smart camera network and an urban traffic control system.

References

[1] C. Müller-Schloer, H. Schmeck, and T. Ungerer, Eds., *Organic Computing – A Paradigm Shift for Complex Systems*. Birkhäuser Verlag, 2011.

[2] J. Hähner, S. Rudolph, S. Tomforde, D. Fisch, B. Sick, N. Kopal, and A. Wacker, "A Concept for Securing Cyber-Physical Systems with Organic Computing Techniques," in *Proc. of ARCS'13 Works.*, 2013, pp. 1–12.

[3] R. Nelsen, *An Introduction to Copulas*. Springer, 1998.

[4] A. J. McNeil, R. Frey, and P. Embrechts, *Quantitative risk management: concepts, techniques and tools*. Princeton University Press, 2005.

[5] N. R. Draper and H. Smith, *Applied regression analysis*. New York: Wiley, 1966.

[6] R. L. Gorsuch, *Factor Analysis*, 2nd ed. Taylor & Francis Group, 1983.

[7] I. T. Jolliffe, *Principal Component Analysis*. Springer, 2002.

[8] C. L. Magee and O. L. de Weck, "Complex System Classification," in *Proc. of INCOSE'04*, 2004.

[9] R. C. Nickerson, U. Varshney, J. Muntermann, and H. Isaac, "Taxonomy development in information systems: Developing a taxonomy of mobile applications." in *Proc. of ECIS'09*, 2009, pp. 1138–1149.

[10] D. Thunnissen, "Uncertainty classification for the design and development of complex systems," in *3rd Predictive Methods Conf.*, 2003.
[11] O. L. de Weck and C. Eckert, "A Classification of Uncertainty for Early Product and System Design," in *Proc. of 16th Int'l Conf. on Engineering Design*, 2007.
[12] E. Cakar, J. Hähner, and C. Müller-Schloer, "Investigation of Generic Observer/Controller Architectures in a Traffic Scenario," in *Beiträge der 38. Jahrestagung der GI*, 2008, pp. 733–738.
[13] S. Tomforde, B. Hurling, and J. Hähner, "Distributed Network Protocol Parameter Adaptation in Mobile Ad-Hoc Networks," in *Selected Papers from ICINCO'11*, ser. LNEE. Springer, 2011, vol. 89, pp. 91–104.
[14] H. Prothmann, J. Branke, H. Schmeck, S. Tomforde, F. Rochner, J. Hähner, and C. Müller-Schloer, "Organic Traffic Light Control for Urban Road Networks," *Int. J. of Autonomous and Adaptive Communications Systems*, vol. 2, no. 3, pp. 203–225, 2009.
[15] U. Richter, H. Prothmann, and H. Schmeck, "Improving XCS Performance by Distribution," in *Proc. of SEAL'08*, ser. LNCS, vol. 5361. Springer, 2008, Inproceedings, pp. 111–120.
[16] H. H. Dam, H. A. Abbass, and C. Lokan, "DXCS: an XCS system for distributed data mining," in *Proc. of GECCO '05*. ACM, 2005, pp. 1883–1890.

Model-based, Adaptive Testing of Organic Computing Systems

Benedikt Eberhardinger
Institute for Software & Systems Engineering, University of Augsburg, Germany
benedikt.eberhardinger@informatik.uni-augsburg.de

***Abstract*—Testing is one of the critical points in the software engineering process, especially in highly complex, autonomic systems. Yet there is no clear concept how to test an Organic Computing System in an appropriate way. The main problem in this field is to handle the self-organizing and adaptive behaviour of those systems. I propose a model-based and adaptive approach to test Organic Computing Systems.**

***Index Terms*—Model-based Testing; Adaptive Testing; Testing; Organic Computing; Multi-agent System**

I. Introduction

Organic Computing (OC) Systems are large, distributed, heterogeneous systems (mostly multi-agent systems), which are aware of their environment and themselves in order to autonomously organize and adapt to achieve certain goals [1]. For that reason, the agents are communicating with other agents to cooperate and interact. This leads to an evolutionary changing system. Due to the fact that the environment and the behaviour of other agents are hard to predict, unforeseeable system states can occur. This fact makes it difficult, but even more necessary, to design convincing tests to detect unintended behaviour of the OC System.

The complexity of testing self-organizing, adaptive systems could be spilt into two major facts: On the one hand, the self-organizing and adaptive characteristics make it tough to design and perform suitable tests. On the other hand, testing distributed, concurrent software is still an awkward challenge [2]. While the issues of testing distributed, concurrent software have been a problem in research for a long time [3], the difficulties of testing OC Systems add a new view. An example is the task to test the aspect of reorganization. Therefore, the test has to deal with the interactions within the OC Systems, because a task could be achieved in many different ways that could radically change by effects of other agents and the environment. For example, it is pretty straight forward to test whether or not a single agent achieves a desired output under a given circumstance. But if the task is allocated to a system of agents which, e.g., could form coalitions for achieving this goal it is rather difficult to say which agent performs correctly or not, which in this example requires to know the responsibilities within the coalitions.

To cope with this complexity, a structured process is needed. The aim is to design a model-based, adaptive approach designed for OC Systems. For that reason, existing techniques for distributed, concurrent software systems are extended for OC Systems. The here presented work of my doctoral dissertation is in an early stage, thus, this paper presents only a crude outline on this topic: First, related work from the field of model-based testing distributed, concurrent systems as well as dynamic symbolic execution for testing is presented in Sect. II. Afterwards the challenges in testing OC Systems are presented in Sect. III. To cope with these challenges my approach is outlined in Sect. IV.

For the further explanations a short, simplified case study of a distributed power management system will be used [4]. The power plants in this system are partitioned into Autonomous Virtual Power Plants (AVPPs). These AVPPs coordinate power plants in order to meet power demands. Each AVPP is represented by an agent and has to fulfil a specific power output to accomplish the global goals of the system. Furthermore an AVPP can recognize a deviation of the global output and react to this in cooperation with the other AVPPs. The demand for each agent is allocated autonomously among the AVPPs. To simplify matters, all AVPPs have the same capabilities, e.g., they all have the same maximum power output in production, same reaction times and same costs.

II. Related Work

In the area of model-based and automatic testing of concurrent, distributed systems different approaches have been introduced and applied. These concepts are partly related to the approach of model-based, adaptive testing of OC Systems. The most important of them will be introduced below.

There has been significant research in the area of model-based testing. Zander et al. [5], Broy et al. [6] and Utting et al. [7] provide an overview of the research in this field. The basic idea of model-based testing is to develop a formal model of the system under test (SUT) and to use this model to generate tests for the system automatically. To obtain this model the requirements and functional models of the SUT are combined to encode the intended behaviour of the system. Therefore, different kinds of models are applied, which describe the tests as well as the intended behaviour [5], [6]. These models have to describe all possible different system transactions. In OC systems it is inpossible to define all possible system transactions. Thus, a model has to be able to be more flexible to support testing of OC Systems. By defining the model in terms of a corridor of correct respectively intended behaviour this flexibility could be gained. My approach is to introduce a goal-oriented behaviour description in the system requirements.

This enables the static model to be more flexible according to the SUT.

Another problem, which has gained special importance in the case of goal-oriented behaviour descriptions, is to generate automatically good test case and the corresponding test oracles. Godefroid et al. [8] addressed this issue with the tool "Directed Automated Random Testing" (DART). The authors combine three main techniques: First the interfaces of a program to its environment are extracted automatically. Afterwards a test driver is generated for all interfaces to perform random tests in order to simulate the environment. Last the output of the program is dynamically analysed how it behaves under random conditions. Based on this results new test paths are generated. By using these techniques, DART is able to test a program that compiles – as a black-box – without writing any test driver. In other words, DART constructs well-formed random inputs for the system, which are used to stress-test the system. On the basis of DART Godefroid et al. [9] developed the SAGE system, which allows white-box fuzz tests. The program is executed, beginning from a defined point in the program code, with a fixed initial input. Afterwards the algorithm of SAGE is gathering input constraints from conditional statements. These collected constraints are used to generate test inputs.

In the field of symbolic execution tests other efficient approaches are made by Rungta et al. [10], Davies et al. [11] as well as Griesmayer et al. [12]. Griesmayer et al. [12] especially take the problem of testing distributed objects into account.

In theory systematic dynamic test generation can lead to full program path coverage. Furthermore it allows the test suite to adapt to the program. But, the problem of these approaches is the lack in scalability, because the techniques could lead to a path explosion (cf. [13]). My presented approach tries to cope with this problem by decomposition.

An alternative way to find the input for the test cases is to use a model checker that generates possible inputs out of a given model. Gargantini et al. [14] showed an approach that is generating test sequences with a model checker. My approach is trying to combine these techniques with the dynamic symbolic execution of tests.

To apply these related work into the approach of testing OC Systems it is necessary to take the fact of distribution and concurrency into account. In the area of testing distributed, concurrent systems a lot of research has been done. Souza et al. [2] gives an overview of the most resent research in this area. These techniques have to be taken into account to be able to test OC Systems adequately, as mentioned in a later section of this paper.

Concluding, in different related areas of research already techniques are developed which could be partly integrated in this approach. Indeed, there is no related work in the area of testing OC Systems. Testing adaptation, self-organization, etc. is an unexplored area, where this work tries to give its contribution.

III. Challenge

In general, testing is "*[the] process of operating a system or component under specified conditions, observing or recording the results, and making an evaluation of some aspect of the system or component*" [15]. The intention of the evaluation is to identify a failure, which is an event that leads to a state of the observed system which does not perform a required function within specified limits [16]. To detect this failure of the observed system, it is necessary to be able to state whether or not the system is in a correct state.

In OC Systems, this is not always straightforward to answer. One approach, made by Schmeck et al. [17] is to separate the global state space of an OC System into a *target space*, an *acceptance space*, a *survival space* and the so called *dead zone*. In this context a failure will occur if the system state is in the *dead zone* and so is "unrepairable" by itself through reconfiguration. Therefore, tests have to check in which space the system is. In this context the following challenges for testing OC Systems could be identified:

- Deal with reconfiguration and adaptivity
- Handle interleaved feedback-loops
- Cope with distribution and concurrency
- Find a clear classification of failures for OC Systems
- Create an appropriate environment for testing
- Define suitable levels of testing OC Systems

A. Reconfiguration and Adaptivity

Reconfiguration and adaptivity allows OC Systems to recover from system states which are not fulfilling the requirements. This situation therefore does not have to be a failure, but it is important that a system which is in a reconfiguration phase still behaves in an acceptable and safe manner. In the example of the AVPPs this case appears if one agent fails to gain the demanded output. But all other agents recognize this deviation and therefore produce together more output as set to each of them. This leads to an achievement of the global goal and therefore no failure despite of a local deviation. So a local deviation (which is not part of the goal) does not lead to a failure, because the system reconfigures and reacts adaptively on this situation.

B. Interleaved Feedback-Loops

The behaviour of the self-organizing and adaptive agents of an OC System could be described by a number of interleaved feedback-loops [17]. As a consequence, it is possible that all agents of the system perform as required, but the whole system fails, e.g., all AVPPs recognize a deviation of the global goal and therefore adapt their own output, because every agent is trying to fix the deviation on its own, the global system overreacts. Consequently every agent performs locally correct by reacting to the new situation, but the system fails with respect to its goal. Thus, one important challenge is to cope with these interdependencies in OC Systems.

C. Distribution and Concurrency

In addition, there are the challenges given by the distribution and concurrency of the systems. Namely deadlocks, livelocks, and data races have to be identified. These are mainly situations where a concurrent usage of a resource leads to a failure.

D. Classification of Failures

Despite the known aspects about possible failures in distributed, concurrent systems and the outlined failures of OC Systems, there is no clear classification of failures for OC Systems. So an important step towards testing OC Systems must be to take this problem into account.

E. Appropriate Environment for Testing

Building up on a classification of failures for OC Systems, it is necessary to consider how to find these possible failures in a concrete system. As a result of the self-organization and the adaptation of an OC System, because of other possibly not controllable agents and the environment of the system, it is difficult to force the system to perform in a way that allows a concrete evaluation. It is indispensable to create an environment for the tested part of the system to generate reproducible results of the evaluation for a concrete requirement. This includes a complete initial configuration with possibly required mock-ups or stubs. For testing the system of AVPPs it is obvious that such a test-suite is inevitable. Mock-ups have to simulate the energy grid, the consumers and other participants to perform suitable evaluations. Furthermore, the adaptivity of a single AVPP should be tested separatly from the other AVPPs to perform reproducible, independent results. For example to test the reaction of an AVPP to a deviation of the global demand and the global production other agents could influence the results and so it is difficult to evaluate this. To perform tests in an appropriate way, a generic approach has to be found for setting up tests in the described way. As a part of this generic approach the differentiation between testing and simulating a system has to be done in a more clear way.

F. Suitable Levels of Testing

Depending on the requirements to be evaluated, tests have to be performed on an appropriate level. For example, tests must be performed for single agents or for groups of agents to observe interdependencies. In the example of the AVPPs there could be one level to test a single AVPP separately, one other to test all controllable AVPPs together and on another level with non-controllable agents. Furthermore there could be several levels of the environment, e.g., the Bavarian energy market, the German energy market, the European energy market and so on. Different levels could lead to different interdependencies and so exhibit different kinds of failure. Common levels of testing are unit tests, integration tests, system tests, and acceptance tests. But OC Systems are, as already shown, different from classical software systems. So the levels have to be redefined and possibly extended according to the dedicated needs.

IV. Approach

To cope with the challenge to test OC Systems adequately, the intention is to develop a model-based, adaptive approach with the following main concepts:

- Decomposition
- Discover sequences of violation
- Model-based design and execution of tests

A. Decomposition

To achieve an adaptive test, techniques of dynamically creating and executing symbolical tests, are used to build up on ([13], [9]). These techniques lead to a high coverage by trying to explore every possible path for a symbolic input automatically. One problem to be solved here is the scalability of this approach, because the number of possible execution paths is exponential in the size of the input [13]. Especially in an OC System this could get problematic, because of its complexity and size. The OC System of the AVPPs, e.g., could represent the whole energy market of Europe, which means an enormous amount of participating agents. For this reason, decomposition plays a significant role. The approach by Steghöfer et al. [18] shows how to use and decompose goal-oriented requirements for monitoring OC Systems. For this purpose a set of constraints for specific agents is defined, which describes the global invariant of the system. This decomposition could be reused in the tests.

Built up on decomposed test cases the scenario has to be extended ”bottom-up“ to even get the possibility to test self-organization, adaptivity and other features of distributed, concurrent systems.

B. Sequences of Violation

Furthermore, it is important to discover the sequences for a possible violation and thus to reduce the size of the paths to be tested. A possibility is to use techniques of model checking to generate sufficient test cases. To enable these process I propose to use techniques build up on model-based testing.

Despite the problem of creating the appropriate test cases, it is important to get a good test oracle for the evaluation. The Restore Invariant Approach (RIA) defines the correct behaviour of an OC System by a corridor of correct behaviour [19]. The corridor of the RIA is based on a global system invariant, which is a conjunction of all constraints of a system. Based on a system trace it is therefore possible to decide whether or not the system is performing correct or has to be reconfigured. This view on correct and incorrect behaviour of an OC System could be adapted as a test oracle for the evaluation. For the example of the AVPPs the system invariant only consist of one constraint, which is namely that the demand equals to the consumption, so every evaluation has to check this characteristic of the system.

C. Model-based Design and Execution of Tests

In this approach models are used to design and execute test cases. For that reason models represent the required behaviour

of the system and its environment. Constraints designed for an OC System could be, as already shown, used to model the required behaviour of the system. Also there is a need of some relies according to the environment to specify the test. A related approach to this modelling is made in the concept of Rely/Guarantee (R/G), which is used by Nafz et al. [20] to formally model and verify OC System. In the case of the AVPP case study the model for the test would be the constraint of balanced production and demand. The rely according to the environment is that the demand would not be bigger than the maximum productivity of the AVPPs. Beside the required behaviour and the relies there is a necessity for structural and behaviour models of the tested components. Therefore, models from the *UML* standard, like activity and class diagrams are obvious to use. These models now could be used to build concrete test cases and a test environment.

D. AVPP Case Study

The global invariant of the AVPP System is that demand is equal to the production of the system, a pretty easy way to break this down to constraints for single agents would be to divide the global demand by the number of AVPPs. Thus on the lowest level each single AVPP could be tested separately from the system. For that reason the test-environment has to mock-up the other agents, which perform for one test the correct and for another the incorrect output. Furthermore there has to be a mock-up, which simulates the consumer. After that the evaluation just has to check whether or not the output of the agent is as high as expected. The expectation is that the agent produces as much output as needed so that the sum of simulated production and its own production equals to the simulated demand. Of course the agent is limited to its maximum productivity, this has to be taken into account for the test cases in form of a rely. In the next case two Agents of the same kind could be tested together in a pretty similar environment to evaluate the interaction, e.g., the reaction on a deviation.

V. Conclusion

Self-organizing and adaptive properties of Organic Computing (OC) systems are challenging research topics with a lot of open issues to be solved (cf. [21]). This paper points out the major challenges in testing OC Systems. My approach is to cope with these problems by using a model-based and adaptive concept. To apply this, useful techniques and processes from the field of testing distributed and concurrent systems [2] will be adapted. This could be combined with work in the field of dynamic creating and executing symbolical tests [13], to achieve a certain kind of adaptivity. The basis for testing is a model, which can be processed. The related work in model-based testing (especially for reactive systems [6]) could give a good initial point.

The aim of the future work is to provide a test suite with an according process to test OC systems.

References

[1] C. Müller-Schloer, H. Schmeck, and T. Ungerer, Eds., *Organic Computing - A Paradigm Shift for Complex Systems*. Springer, 2011.

[2] S. R. S. Souza, M. A. S. Brito, R. A. Silva, P. S. L. Souza, and E. Zaluska, "Research in concurrent software testing: a systematic review," in *Proc. Wsh. Parallel and Distributed Systems: Testing, Analysis, and Debugging*, ser. PADTAD '11. ACM, 2011, pp. 1–5.

[3] C.-S. D. Yang, "Program-based, structural testing of shared memory parallel programs," Ph.D. dissertation, University of Delaware, 1999.

[4] G. Anders, F. Siefert, J.-P. Steghfer, H. Seebach, F. Nafz, and W. Reif, "Structuring and controlling distributed power sources by autonomous virtual power plants," in *Proc. of the IEEE Power and Energy Student Summit*, ser. PESS '10. IEEE, 2010.

[5] J. Zander, I. Schieferdecker, and P. J. Mosterman, Eds., *Model-Based Testing for Embedded Systems*. CRC Press, 2012.

[6] M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner, *Model-Based Testing of Reactive Systems*, ser. LNCS. Springer, 2005.

[7] M. Utting, A. Pretschner, and B. Legeard, "A taxonomy of model-based testing approaches," *Softw. Test. Verif. Reliab.*, vol. 22, no. 5, pp. 297–312, Aug. 2012.

[8] P. Godefroid, N. Klarlund, and K. Sen, "Dart: directed automated random testing," in *Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation*, ser. PLDI '05. New York, NY, USA: ACM, 2005, pp. 213–223.

[9] P. Godefroid, M. Y. Levin, and D. A. Molnar, "Automated whitebox fuzz testing," in *Proc. 16th Annual Network and Distributed System Security Symposium*, ser. NDSS '08. Internet Society, 2008.

[10] N. Rungta, E. G. Mercer, and W. Visser, "Efficient testing of concurrent programs with abstraction-guided symbolic execution," in *Proc. 16th International SPIN Workshop on Model Checking Software*. Springer-Verlag, 2009, pp. 174–191.

[11] M. Davies, C. Psreanu, and V. Raman, "Symbolic execution enhanced system testing," in *Verified Software: Theories, Tools, Experiments*, ser. LNCS, R. Joshi, P. Mller, and A. Podelski, Eds. Springer, 2012, vol. 7152, pp. 294–309.

[12] A. Griesmayer, B. Aichernig, E. B. Johnsen, and R. Schlatte, "Dynamic symbolic execution for testing distributed objects," in *Proc. 3rd International Conference on Tests and Proofs*, ser. TAP '09. Springer-Verlag, 2009, pp. 105–120.

[13] C. Cadar, D. Dunbar, and D. Engler, "Klee: unassisted and automatic generation of high-coverage tests for complex systems programs," in *Pro. 8th USENIX Conf. Operating systems design and implementation*, ser. OSDI'08. USENIX Association, 2008, pp. 209–224.

[14] A. Gargantini and C. Heitmeyer, "Using model checking to generate tests from requirements specifications," in *Proc. 7th European software engineering conference held jointly with the 7th ACM SIGSOFT international symposium on Foundations of software engineering*, ser. ESEC/FSE-7. Springer-Verlag, 1999, pp. 146–162.

[15] IEEE, *IEEE Standard Glossary of Software Engineering Terminology*, IEEE Std. 610.12-1990.

[16] ——, *IEEE Standard Classification for Software Anomalies*, IEEE Std. 1044 - 2009.

[17] H. Schmeck, C. Müller-Schloer, E. Çakar, M. Mnif, and U. Richter, "Adaptivity and self-organization in organic computing systems," *ACM Trans. Auton. Adapt. Syst.*, vol. 5, no. 3, pp. 10:1–10:32, Sep. 2010.

[18] J.-P. Steghöfer, B. Eberhardinger, F. Nafz, and W. Reif, "Synthesis of observers for autonomic evolutionary systems from requirements models," in *Proc. 13th IFIP/IEEE Symposium on Integrated Network and Service Management*, ser. IM '13. IEEE Computer Society, 2013.

[19] F. Nafz, H. Seebach, J.-P. Steghöfer, G. Anders, and W. Reif, "Constraining self-organisation through corridors of correct behaviour: The restore invariant approach," in *Organic Computing A Paradigm Shift for Complex Systems*, C. Müller-Schloer, H. Schmeck, and T. Ungerer, Eds. Springer, 2011, pp. 79–93.

[20] F. Nafz, J.-P. Steghfer, H. Seebach, and W. Reif, "Formal modeling and verification of self-* systems based on observer/controller-architectures," in *Assurances for Self-Adaptive Systems*, ser. LNCS, J. Cmara, R. Lemos, C. Ghezzi, and A. Lopes, Eds. Springer, 2013, vol. 7740, pp. 80–111.

[21] C. Nguyen, A. Perini, C. Bernon, J. Pavn, and J. Thangarajah, "Testing in multi-agent systems," in *Agent-Oriented Software Engineering X*, ser. LNCS, M.-P. Gleizes and J. Gomez-Sanz, Eds. Springer, 2011, vol. 6038, pp. 180–190.

FESAS: A Framework for Engineering Self-Adaptive Systems

Christian Krupitzer
University of Mannheim, Germany
christian.krupitzer@uni-mannheim.de

***Abstract*—The complexity and size of information systems are growing, resulting in an increasing effort for maintenance. Autonomic Computing, Organic Computing or self-adaptive systems (SAS) that autonomously adapt to environment changes or changes in the system itself (e.g. disfunction of components) can be a solution. So far, the development of a SAS is tailored to its specific use case requirements. The creation of frameworks with reusable process elements and system components is often neglected. However, with such a framework developing SAS would become faster and less error prone.**

This work addresses this gap by providing a framework for engineering SAS. The framework is based on model-driven engineering and a service-oriented architecture. At the development stage, a design model is transformed into a system model. During runtime, this system model is mapped to a concrete system build of various heterogeneous devices with functionalities. The functionalities of the devices is represented as services. A reference architecture and a component library support this mapping and enable the deployment of an adaptation logic tailored to the system's use case. This reduces engineering time through a generic process with reusable elements.

***Index Terms*—Self-Adaptive Systems; Software Engineering; Model-Driven Engineering; Service-Oriented Architecture; Pervasive Computing; BASE Middleware**

I. Introduction

Recently, the complexity and size of information systems are increasing, especially in systems that are composed of heterogeneous devices, such as pervasive environments. This results in higher effort for maintenance. A solution is the design of self-adaptive systems (SAS) that autonomously perform tasks, known as the self-* properties (for a description of the self-* properties see [10]). A formal definition is given by Oreizy *et al.* [16]:

> Self-adaptive software modifies its own behavior in response to changes in its operating environment. By operating environment, we mean anything observable by the software system, such as end-user input, external hardware devices and sensors, or program instrumentation.

The focus of this proposal is self-adaptation, which is the modification of system behavior in response to changes in its operating environment. Hence, self-adaptation is a basic but powerful mechanism for enabling Autonomic Computing, which is described in analogy to the autonomous nervous system as systems that perform tasks autonomously [10]. Another related research area of SAS research is Organic Computing. In Organic Computing concepts to achieve controlled self-organization are sought after [18].

The remainder of the proposal is structured as follows: in Section II the problem of missing reusability in the engineering processes for SAS is defined. The relevance of the problem is stated in Section III. Some approaches to the problem are given in Section IV. The framework is explained in Section V. Section VI presents challenges and the roadmap for the development of the framework.

II. Problem Definition

SAS research mainly addresses approaches for achieving self-adaptation by designing systems tailored to specific use cases. Design decisions related to the specific needs of the use cases complicates reusability of the resulting artifacts.

The creation of frameworks for designing and engineering SAS is often neglected [2]. Especially, there is a lack regarding generic process elements for SAS design and engineering [3]. Furthermore, a library of reusable components like control structure elements would support the development of adaptation logic for SAS [3].

Integrating the process definitions and a library of components into a framework enhanced with tools would simplify the development of SAS and result in faster and less error prone development.

III. Problem Relevance

In general, there are two approaches existing for adaptation of software: parameter adaptation and compositional adaptation [12]. Parameter adaptation is concerned with adapting the system's behavior through changing various parameters. The focus in this work is on compositional adaptation, which is described as exchange of components [12]. Additionally the compositional adaptation is divided in an internal approach, where application and adaptation logic are integrated, and an external one with a dedicated adaptation logic [17]. In this work the external adaptation approach is addressed. Different approaches for systems that are able to adapt their behavior via compositional adaptation exist. These are presented in this section.

Architecture-based approaches have in common that specific architectural components or layers control and execute the adaptation (e.g. layer approach [11], adaptation manager [16], and the Rainbow framework [6]). Other approaches are based on models for controlling adaptation (e.g. Models@Run.time

[14], architectural models [5], and adaptation models [20]). For the composition of artifacts, service-based techniques are beneficial. Here, services can be composed in different chronological orders and service implementations are exchangeable for handling changing (environmental) requirements. Therefore, service architectures are commonly used for simplifying the compositional adaptation (e.g. [4], [7], [13]).

Organic Computing [15] or Autonomic Computing [10] are derived from principles of adaptation found in biology, physics, or chemistry that are transfered to information technology.

However, most systems based on the above mentioned approaches are lacking the ability of generalization as well as reusable and well-defined processes and components for design and implementation [3]. Furthermore, the authors mostly focus on one of the categories (architecture-based, model-based, or service-based) instead of combining them in order to benefit from their advantages – e.g. the abstraction of service orientation and ease of model-driven design – while minimizing the respective drawbacks – e.g. the fixation on specific use case requirements. Additionally, the focus is mainly on addressing single self-* properties.

IV. Related Work

Different approaches already addressed issues in software engineering for SAS. Seebach *et al.* presented software engineering guidelines for self-organizing resource-flow systems [19]. The guidelines are rather generic and applicable in many different cases. Concrete and reusable elements are missing.

Frameworks are presented by Hallsteinsen *et al.* [8], Weyns, Malek and Andersson [21], and Menascé *et al.* [13]. These approaches address some of the issues that should be provided within the FESAS project.

Hallsteinsen *et al.* [8] focus in the MUSIC project on self-adapting applications in ubiquitous environments, a similar setting like in the FESAS project. Nevertheless in FESAS there will be requirements engineering activities integrated which is not explicitly mentioned in MUSIC.

Weyns *et al.* [21] provide with the FORMS reference model a meta-level description of SAS. Their notation can be used for comparing and analyzing different alternatives regarding self-adaptive elements. Nevertheless, concrete methods, tools, or practices are missing.

SASSY is a model-driven framework for supporting the development of systems in dynamic environments, e.g. SAS [13]. The focus here is on self-architecting software, whereas FESAS is concentrated on the adaptation mechanisms.

An approach to adaptation in pervasive environments, specialized in adaptation of distributed systems, offers PCOM [9]. PCOM is build on top of the BASE middleware [1], which should be used in the FESAS project, too, and could be partly integrated.

What is missing so far is mainly the reusability of concrete components and system parts as well as tools for supporting designers and engineers. This work wants to address the gap.

V. Approach

The goal is to create a generic framework for engineering SAS with reusable processes and components as it has been identified as gap in the research landscape of SAS engineering [3]. The term ”generic” means that the framework should enable the construction of self-adaptive systems with different requirements and in different settings. Therefore, it will be necessary that designers can use specific tools that are able to abstract from the low-level requirements of specific systems and offer a high-level abstraction for parts of the system requirements. These high-level requirements will relate to typical requirements for SAS. The framework will offer solutions to these high-level requirements and determine autonomously the system infrastructure needed and build these systems. Low-level requirements that are specific to a special SAS must be implemented separately, but – once implemented – will be integrated into the SAS autonomously.

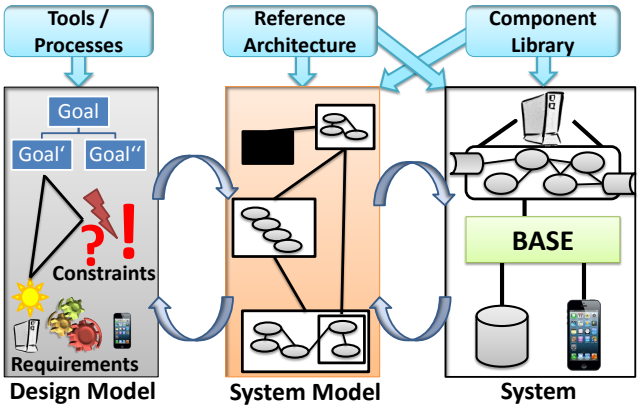

Fig. 1: Framework for Engineering Self-Adaptive Software

Figure 1 shows the conceptual design of the framework. The framework provides tools for designing the system, a reference architecture for SAS and a library with reusable components that are used for building the system. The design is based on model-driven engineering. In the design phase, requirements, goals, and constraints are captured in a design model using a SAS tailored requirements engineering approach [2]. The design model can automatically be transformed into a run-time system model.

The system model offers an initialization of the system based on elements of a component library, which are attached to a reference architecture. The reference architecture consists of the BASE middleware [1] for communication between the different elements and containers for self-adaptive elements which are filled in at run-time with components of the component library.

The reference architecture is designed in a general way in order to apply to many different use cases. Besides self-adaptation, the system has to handle different, heterogeneous devices. Context-awareness is essential because of the necessity to respond to environment changes with self-adaptive

mechanisms. Generic components like self-* properties containers, control loop elements (e.g. MAPE cycle [10]), context manager, communication infrastructure, adaptation manager, and Quality of service (QoS) manager (for non-functional requirements) are part of the reference architecture.

The elements needed for self-adaptation are arranged in one or more adaptation logic components. All elements are in the component library. The distribution of the elements is determined by the framework and based on rules. The functionality of the system is captured in the managed resources. These consist of BASE services, which are managed through the adaptation logic elements. Figure 2 shows the interplay of the adaptation logic and the managed resources.

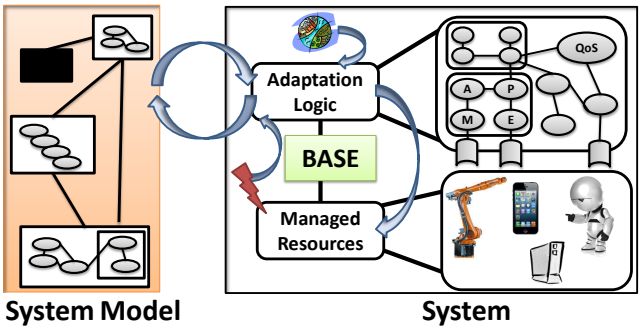

Fig. 2: Connection of Adaptation Logic and Managed Resources in the System

The system architecture is based on the service-oriented architecture (SOA) principle. Communication between the different system artifacts is facilitated by the BASE middleware [1]. The BASE middleware supports the use of services on remote devices. It has been developed to the needs of pervasive environments with many heterogeneous devices. Beside the location of services (no difference in the usage of local or distributed services), BASE is able to abstract the communication in a way that different transport mediums like W-LAN, Ethernet or Bluetooth are supported and the middleware decides autonomously on the suitable one [1]. Each element of the component library, as well as additional functionalities, are modeled as BASE services. With the BASE middleware, the system is modeled as a connection of services and compositional adaptation as exchange of services. In Figure 2 circles represent BASE services, connections between circles represent data exchange via a network connection. The exchange is managed by the BASE middleware. Additionally, sensors are represented by the rectangles with the half-cycled sides top and down.

The creation and adaptation of the system is done in accordance with the design model. Once the system is running, new detected requirements and constraints for the system are used for refinement of the system model (and verifying it against the design model), which can activate the reorganization of system components. The reorganization is determined by the adaptation logic and triggered by environment changes or corruption of system components as shown in Figure 2. Through this reorganization, compositional adaptation is achieved which enables adaptation in changing environments.

Finally, the engineering framework will provide tools for engineering SAS. With these tools and the library of reusable components, models for capturing requirements, goals, and constraints can be created and transformed during run-time into an infrastructure of services.

A specific degree of decentralization results from the heterogeneity of devices. So special attention is laid on decentralized vs. centralized control structures. Modeling an explicit control structure is an important issue in engineering SAS [2]. Additionally, how to implement and maintain the self-* properties according to the degree of decentralization is considered by the framework. The issues mentioned influence the creation of the adaptation logic.

VI. Challenges and Future Work

For achieving the goal, various research challenges have to be tackled:

- Analyzing existing approaches regarding their strengths, weaknesses and similarities.
- Development of a design model language which captures requirements, goals and constraints. Furthermore, a language for developing system models is needed.
- Implementation of a transformer for transforming the design model into a run-time system model and system configuration.
- Derive a catalog with common reusable components for SAS. Special focus will be on control mechanisms for the adaptation and self-* properties.
- Propose a reference architecture for SAS.
- Introduce a software engineering approach for constructing SAS based on the reference architecture and the catalog of reusable components. Therefore, software engineering processes must be tailored to SAS and a new requirements engineering approach for requirements capturing during design as well as run-time must be developed.
- Evaluation of the approach: The systematic evaluation of the framework must be ensured with a methodology.

Right now, a prototype for modeling systems with the component library is under development. Next to a catalog of components for SAS, the components will be implemented as BASE services using Java. Additionally, a modeling syntax based on UML will be designed and a transformer for initializing the system based on the system model will be implemented. Typical components are elements for controlling the adaptation to environment changes. Therefore, the MAPE-K cycle will be implemented and the components for Monitoring, Analyzing, Planning and Executing as well as a Knowledge component will be implemented as BASE services. Further components are sensor interfaces, sensor fusion components (for supporting the monitoring component) or a QoS manager for controlling non-functional requirements.

Furthermore, an evaluation of SAS use cases and scenarios will be done. In this evaluation, factors that induce a specific

degree of (de)centralization will be analyzed and patterns for decentralization will be implemented for the framework. Using these factors, the framework can automatically determine the need of decentralization based on the design model and implement it accordingly in the adaptation logic.

One of the challenges is the modeling of requirements, goals and constraints in an appropriate modeling language. Due to the dynamics in the environment of SAS and the resulting uncertainty of the environment during design time, the requirements engineering of SAS needs a special handling [2]. Therefore a suitable requirements engineering process must be constructed.

Afterwards, a component for transforming the design model into a run-time model and its system configuration must be implemented. Special focus will be on decentralization, control loop aspects, and the degree of user integration. The evaluation of the framework must be ensured. Therefore, a methodology will be developed or an existing one will be adjusted. Due to the requirements that the framework should be as generic as possible, it sounds reasonable that the evaluation should be done in different scenarios. As such scenarios, traffic management, smart logistics and production facilities, or software for robots can be conceivable.

If feasible, existing approaches and components will be integrated into the framework. Especially for the model-driven part of the framework, different solutions are present that could be integrated (e.g. [4], [7], [13]). So the focus of the work will be on the requirements specifications, software engineering processes, and the composition of the resulting system with reusable components.

Remarks

This work is supervised by Prof. Dr. Christian Becker, University of Mannheim, Germany. The FESAS project's homepage is available at:

`http://fesas.bwl.uni-mannheim.de`

REFERENCES

[1] BECKER, C., SCHIELE, G., GUBBELS, H., AND ROTHERMEL, K. BASE - a Micro-broker-based Middleware for Pervasive Computing. In *IEEE International Conference on Pervasive Computing and Communications* (2003), IEEE, pp. 443–451.

[2] CHENG, B. H. ET AL. Software Engineering for Self-Adaptive Systems: A Research Roadmap. In *Software Engineering for Self-Adaptive Systems, LNCS 5525*. Springer, 2009, pp. 1–26.

[3] DE LEMOS, R. ET AL. Software Engineering for Self-Adaptive Systems: A Second Research Roadmap. In *Software Engineering for Self-Adaptive Systems II, LNCS 7475*. Springer, 2013, pp. 1–32.

[4] DI NITTO, E., GHEZZI, C., METZGER, A., PAPAZOGLOU, M., AND POHL, K. A journey to highly dynamic, self-adaptive service-based applications. *Automated Software Engineering 15*, 3-4 (2008), 313–341.

[5] FLOCH, J., HALLSTEINSEN, S., STAV, E., ELIASSEN, F., LUND, K., AND GJORVEN, E. Using architecture models for runtime adaptability. *IEEE Software 23*, 2 (2006), 62–70.

[6] GARLAN, D., CHENG, S.-W., HUANG, A.-C., SCHMERL, B., AND STEENKISTE, P. Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure. *Computer 37*, 10 (2004), 46–54.

[7] GEIHS, K., REICHLE, R., WAGNER, M., AND KHAN, M. U. Modeling of Context-Aware Self-Adaptive Applications in Ubiquitous and Service-Oriented Environments. In *Software Engineering for Self-Adaptive Systems, LNCS 5525*. Springer, 2009, pp. 146–163.

[8] HALLSTEINSEN, S., GEIHS, K., PASPALLIS, N., ELIASSEN, F., HORN, G., LORENZO, J., MAMELLI, A., AND PAPADOPOULOS, G. A development framework and methodology for self-adapting applications in ubiquitous computing environments. *Journal of Systems and Software 85*, 12 (2012), 2840–2859.

[9] HANDTE, M., SCHIELE, G., MATJUNTKE, V., BECKER, C., AND MARRÓN, P. J. 3PC: System Support for Adaptive Peer-to-Peer Pervasive Computing. *ACM Transactions on Autonomous and Adaptive Systems 7*, 1 (2012), Article 10.

[10] KEPHART, J. O., AND CHESS, D. M. The Vision of Autonomic Computing. *Computer 36*, 1 (2003), 41–50.

[11] KRAMER, J., AND MAGEE, J. Self-Managed Systems: an Architectural Challenge. In *ICSE Workshop on the Future of Software Engineering* (2007), pp. 259–268.

[12] MCKINLEY, P., SADJADI, S., KASTEN, E., AND CHENG, B. Composing adaptive software. *Computer 37*, 7 (2004), 56–64.

[13] MENASCE, D., GOMAA, H., MALEK, S., AND SOUSA, J. SASSY: A Framework for Self-Architecting Service-Oriented Systems. *IEEE Software 28*, 6 (2011), 78–85.

[14] MORIN, B., BARAIS, O., JEZEQUEL, J.-M., FLEUREY, F., AND SOLBERG, A. Models@Run.time to Support Dynamic Adaptation. *Computer 42*, 10 (2009), 44–51.

[15] MÜLLER-SCHLOER, C., SCHMECK, H., AND UNGERER, T., Eds. *Organic Computing - A Paradigm Shift for Complex Systems*, 1st ed. Springer, 2011.

[16] OREIZY, P., GORLICK, M., TAYLOR, R., HEIMHIGNER, D., JOHNSON, G., MEDVIDOVIC, N., QUILICI, A., ROSENBLUM, D., AND WOLF, A. An Architecture-Based Approach to Self-Adaptive Software. *IEEE Intelligent Systems 14*, 3 (1999), 54–62.

[17] SALEHIE, M., AND TAHVILDARI, L. Self-Adaptive Software: Landscape & Research Challenges. *ACM Transactions on Autonomous and Adaptive Systems 4*, 2 (2009), Article 14.

[18] SCHMECK, H., MÜLLER-SCHLOER, C., CAKAR, E., MNIF, M., AND RICHTER, U. Adaptivity and Self-organisation in Organic Computing Systems. In *Organic Computing - A Paradigm Shift for Complex Systems*, C. Müller-Schloer, H. Schmeck, and T. Ungerer, Eds., 1st ed. Springer, 2011, pp. 5–37.

[19] SEEBACH, H., NAFZ, F., STEGHOFER, J.-P., AND REIF, W. A Software Engineering Guideline for Self-Organizing Resource-Flow Systems. In *IEEE International Conference on Self-Adaptive and Self-Organizing Systems* (2010), IEEE, pp. 194–203.

[20] VOGEL, T., AND GIESE, H. Adaptation and Abstract Runtime Models. In *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems* (2010), ACM Press, pp. 39–48.

[21] WEYNS, D., MALEK, S., AND ANDERSSON, J. FORMS: a FOrmal Reference Model for Self-adaptation. In *International Conference on Autonomic computing* (2010), ACM Press, pp. 205–214.

Self-Adaption of Multi-Sensor-Systems with Organic Computing Techniques

Martin Jänicke
Intelligent Embedded Systems Lab
University of Kassel
Kassel, Germany
jaenicke@uni-kassel.de

***Abstract*—This excerpt proposes a novel approach towards the autonomous integration of new sensors into a running multi-sensor-system. Considered systems have the main purpose of classifying input data and using adaptions to respond to changes in the environment. For high classification rates the system is supposed to use a discriminative classifier. To cover newly available data, the input space has to be expanded and the system model has to be adapted. The change is realized by combining the main classifier with a generative classifier and using techniques from the area of semi-supervised learning for training. Inferring labels for a new model is the most challenging step, before the evaluation reveals gain/loss compared to the system used before. During this step, selection strategies from the field of active learning are regarded as beneficial. In the early stage of the development process, the appearance of samples in clusters and a correspondence between class label and data cluster are two major assumptions.**

***Index Terms*—classification, self adaption, semi-supervised learning, active learning, organic computing**

I. Introduction

Sensors and sensor-systems are nowadays distributed in various different places of our daily routine. They are used for a lot of different tasks, reaching from healthcare and medical supervision, via activity recognition to technical measurement systems. Especially in the former applications, body-worn sensors play a big role. In most cases, these are attached directly to the body or are at least very close to the body, like smartphones. Another category of sensors are embedded in our environment as measurement or observation instruments, e.g. microphones or cameras. Usually, both types of systems are used to measure specific physical effects or changing conditions. Those measurements are afterwards fed into a classification system that provides a class estimation for the given data. Daily applications are furthermore very dynamic in the sense that every time a person moves or switches places, the availability of sensors can change, too. A question arising in this context is, how to make use of new sensor data that becomes available, as soon as a place with (intelligent) sensing infrastructure is available?

The main focus of the proposed ideas is to address the problem of automatic adaption of sensor-systems fulfilling classification tasks. This should be approached from two directions: One is the addition of a new sensors to the system, which has to be evaluated. In the end, a decision regarding the usage (or avoidance) of the new data should be drawn. Second is the observation, that a system "loses" a sensor (e.g. by leaving its sending range) as input device and thus has to adapt to that new condition.

Ideas collected so far have two major assumptions, which have to be met. The first is the appearance of data in clusters of the input space. The second is a correspondence between classes and clusters: One cluster belongs to one class only. Although these assumptions are only met for artificial data, they ease up the development process at the beginning.

The problem of acquiring, preprocessing and aggregating sensor-information before classification is well understood and has some quite established paradigms. In contrast, the integration of new input sources into a multi-sensor-system during runtime is not. So far, only few systems allow the addition of further sensors while running, often they have to be added manually or can only be used after restarting the system. For ease of use and furthermore having a transparent functionality for users, both steps are not desirable. Another assumption in this context is, that "a sensor just happens". This means, that the necessity for a new sensor is neither measured nor evaluated. The motivation behind using such a new input source, is having more information (= more input sources) available than before and benefit from it. A challenge directly arising from this concept is the question, how to use the existing knowledge base for creating a new model, which incorporates the new data? The answer consists of multiple smaller tasks that have to be solved, which are partly presented throughout this abstract.

Reactions following the newly created model or different strategies to handle such a system with respect to a given metric (e.g. costs or computing time) are not part of the proposed resarch. Also security related questions, like compromised sensor-data or the identification of attack-vectors for such adaptive systems are far fetched, as more fundamental problems, i.e. regarding the adaption process itself, have to be solved first. The goal of this work is to create a methodological ground for self adapting multi-sensor systems, that bases upon mathematical relations, motivated by proof instead of heuristics. A very good proof-of-concept using the latter is given in [1].

The remainder of this text starts with a brief overview of related work and a description of planned techniques, before discussing some ideas to autonomously create a valid classifier from an existent one. Finally, an outlook towards design ideas,

planned investigations and a case study is given. A conclusion is left out on purpose, as it can not be formulated at the early stage of this project.

II. Related work

The field of related work is covered shortly, as the current stage of the dissertation does not include an exhaustive literature research yet. [1] can be seen as a direct predecessor of this thesis, where the basic ideas were successfully implemented as a proof of concept. However, a lot of heuristics were used to create a decision tree as discrimnative classifier in corporation with a simple clustering algorithm as generative counter-part. Also closeley related is [2], where the usage of a new sensor requires a system reconfiguration and is based on assumptions about user behaviour.

Widely available are systems that use a fixed number of sensors, but have to be reconfigured, if not additionally restarted, in order to use another sensor, that is made available. Runtime-adaption of such systems, regarding a change in input-space-dimensionality are currently not available. Some publications scope on the problem of using techniques from the field of semi-supervised learning for activity recognition (e.g. [3]), but a lack that should be overcome is the fixed dimensionality of the input space. In other words, the main interest lies in dynamic that occurs in the input spaces, not systems with a dynamic architecture.

III. Expanding N sensor dimensions

Supposing a sensor-system with N dimensions as a starting point, a first question is: Which extensions are possible? The generic answer is, additionally occuring sensors can have up to M dimensions, but until noted otherwise, this text argues with the simplified expansion of only one dimension.

So a first challenge is to answer the question: How can another dimension be used by the system? In the most primitive case, a sensor that becomes available to the system carries only one-dimensional information. Followingly, the system aims at expanding the dimensionality of its input space to $N+1$. Up to this point, labeled data is only available for the N dimensions observed so far. The new data has to be adapted somehow, in a way that labels are inferred from the lower dimensional input space.

A simple approach for this uses the projection of new samples into the N dimensional space. For further explanations this set is called S_{Sel}. With the assumption that clusters have only one corresponding class, samples with known labels can be used to label elements of S_{Sel}. If S_{Sel} is afterwards projected back into its originating ($N+1$ dimensional) space, labeled information for creating a classifier has become available there. A very important question still remaining, addresses the selection strategy that is used to label projected samples from S_{Sel} in the original N dimensional input space. Furthermore, the same question holds for samples, which are observed after this process: How can they be marked reliably in $N+1$ dimensions, if only few observations have a class label? The answer is expected to be found in the area of "Semi-Supervised Learning" (cf., e.g., [4]) with selection strategies from the field of "Active Learning" (cf., e.g., [5]) as described in the next section. As a result of these techniques, the system should be able to work autonomously, without user-interaction.

IV. Proposed techniques

To achieve optimal classification results, the main classifier of the overall system should be a discriminative classifier. It works on the original N dimensions and should, after the aforementioned transformation, also work on the adapted $N+1$ dimensional input space. After receiving a "sufficient number" (see below) of samples in the latter mentioned input space, it is advisable to create a generative classifier. For example, when using a "Gaussian Mixture Model" (GMM) as generative classifier, the components can be initialised via Variational Bayesian Inference [6] as shown in [7]. Using the method of projecting data from the new input space into the old one, those points can be labeled easily. One method would be to select a set of data points P_{High} with all elements having a high density, project them into the N dimensional space, label them (via the cluster – class correspondance assumption) and label the corresponding cluster in $N+1$ dimensions after back-projection of P_{High}. Obviously, the labeling of clusters in the $N+1$ dimensional space is a non-trivial task, but more on that later. The result of this whole process is a new generative classifier in the higher dimensional input space. It is expected, that such a classifier can not fulfill the requirement of a high classification rate. This is the main argument for using discriminative techniques for the classification task itself. Training such a classifier is another challenge of the process. The current idea is to observe data points, that are labeled by the generative classifer, and use them as training-input for the discriminative classifier. This semi-supervised approach should allow the creation of a discrimative classifier by using a generative classifier for label-provision. After adapting the system, the changes have to be evaluated for "some time" (see below) and a decision has to be formed, whether to use the new model – or discard it and continue using the old model. So far for the direction of adding a sensor to the system. The other direction (a system losing a sensor) requires at least an observation by the system, which recognizes the lack of information in one dimension (e.g. missing values or unplausible values). A possible solution to this problem seems to be quite similar to Obsoleteness Detection as described in [8], [9]. Nevertheless, this procedure has at least one unknown variable in common with the system adaption described before: The best choice of "some time" for evaluation is unknown. As a direct consequence of the proposed steps, the following questions can be formulated and have to be studied carefully:

1) What is a "sufficient number" of samples, before initialising the new classifier?
2) Which other generative classifier models can be used to replace the GMM-based approach?
3) What other selection strategies should be investigated as alternatives to selecting data points of high density?

4) Which measures should be used to evaluate the gain/loss of the new model, compared to the old one?
5) How long is "some time", which should be used for evaluating an adapted model?

Up to now, these questions are unanswered, as this work is in a very early stage of development. However, it is important to divide them into subtasks and find solutions and alternatives for the mentioned points. Another area of research that will be beneficial for this work is the usage of established selection strategies for samples from the field of Active Learning. Quite some effort was put in the development of different selection strategies in this domain by other members of the OC group [10], [11].

V. Outlook

After touching some ideas towards a self-adapting multi-sensor-system, multiple questions are still to be answered. Despite the tasks identified in the section before, the following challenges relate to basic assumptions made so far:

- Is it possible to losen the cluster – class correspondance assumption? Usually, non-artificial datasets do not follow this easy-to-model approach. Followingly, it is desirable to develop methods that work, while facing the challenges of real world datasets.
- In most applications rather specialized approaches are chosen to solve the underlying problem. In the scope of this theme, an interesting challenge is to integrate knowledge about a certain problem into the system. Desirable, but yet unknown, is a standardized procedure, that would allow the usage of prior knowledge within the architecture that is described below.
- How does a system cope with the addition of more than one sensor dimension? Sensors typically measure an effect in multiple degrees of freedom, computing more than one feature along each degree. So the assumption of only adding a single dimension to the input space is met by artificial datasets only, whereas real world datasets are expected to add more than one dimension at once, when a system starts using a new sensor.
- When multiple dimensions become available to the system, one way to cope with them would be to add only one dimension at a time, evaluate, add the next one and so on. A downside of this approach would be the time necessary, to fully use (add, label, evaluate) all new dimensions.
- Although only parametric models are proposed, is it possible/useful to use non-parametric models as well? It could be possible, that exchanging some methods would increase the overall performance of the system.
- So far, the suggested solutions use probabilistic techniques for modelling. Quite some effort should be used to find ways, in which this can be beneficial.
- Can timing-information be helpful? Suppose a situation, where a system with a new sensor is under evaluation and knowledge about a similar sensor, that was used some time ago, can be used to reach a decision (accept or discard) more quickly.

Comparing models, evaluating classifiers, determining differences between distributions or between samples and distributions requires appropriate measures, that are available (cf. Mahalanobis-distance, Symmetric Kullback-Leibler divergence, Hellinger-distance) – or have to be developed within the scope of this dissertation. Further work will be put in the development of a modular software realization, to conduct experiments. This will be based on datastructures and algorithms from a framework, which is currently developed by the Organic Computing group of the Intelligent Embedded Systems lab at the University of Kassel.

A prototypical system is going to be built in a layered structure, with the established observer/controller pattern as recurring structure within each level. Figure 1 shows a possible structure for this.

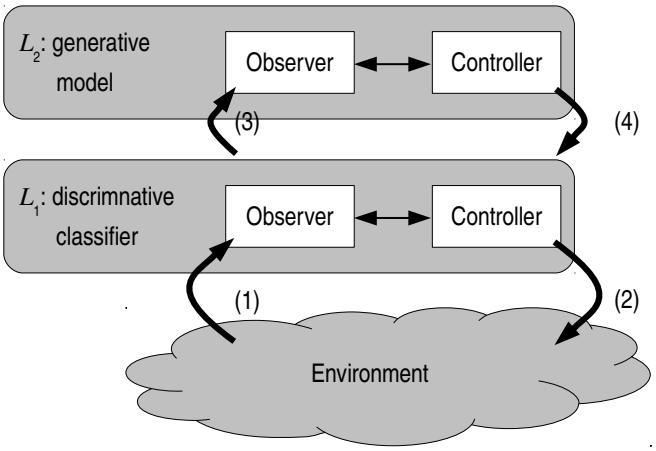

Fig. 1. Layered architecture of a proposed system.

The graphic visualizes the connection between the two layered system and the environment it is interacting with. Within each layer, an observer and a controller are contained. The basic idea is, to let one part focus on and work with incoming data (observer), while the other part interacts with everything outside the current layer (controller). To realize this division both components are able to communicate with each other. The proposed design uses two layers, but communication only takes place between observer and controller of the same level.

In a top down approach, sensor values are obtained by the system as a whole, but when looking into the details, it is mostly the discriminative classifier in the first layer (L_1), that uses those measurements. After several preprocessing steps, they are classified (1) and the classification result might trigger some reaction in the environment (2). This short description obviously involves communication between observing and controlling parts of the lowest layer. However, as this work focusses mainly on the methods, that classify data and adapt the system in use, outputs of the controlling part (2) can be neglected. In other words, they can be seen as base for different reaction strategies – but investigating concrete actions or strategies based on a triggered reaction lies not within the scope of this work.

The second layer (L_2) has the purpose of helping during the automatic training of a new classifier and monitoring L_1. To be more precise, L_2 monitors the adapted discriminative classifier in the first layer L_1 and compares it to the old model used in L_1 for a certain time (3). If the new model proves to be better, it replaces the old one (4). If the new model receives a worse evaluation than the original model, it is discarded and the old model is used instead (4).

Intentionally avoided was the theme of feature selection, to achieve optimal classification results. Being a very interesting domain of its own, it is not planned to focus on this theme while working on the described challenges. Equally important is the question for synchronization mechanisms. When receiving data from a new sensor, how can be ensured, that the incoming data stream is synchronized with the already existing streams? Autonomuous approaches using a synchronization on certain events in the data stream were realized as proof of concept in [1]. Advantageous of that proposal is the fact, that it does not rely on the synchronization of devices' internal clocks. An obvious downside is the accuracy, which is limited to the detection mechanisms, that analyze the datastreams. Followingly, deviating ways and synchronization methods should be investigated.

Anomaly detection mechanisms for GMMs were eveloped quite well as proposed in [12]. Those knowledge is expected to be useful in the context of creating resilient generative models, that model observed data as good as possible. However, alternative methods and modelling techniques have to be considered and compared to this familiar approach.

Most important and not to be neglected is a thorough research for related work and state of the art publications, that offer established procedures for the various challenges of this dissertation theme.

For the evaluation of techniques and system architecture, a case study in the field of activity recignition is planned. Activity regcognition offers a variety of chances, to proof the feasability of the suggested ideas: While attempting to detect a user's action, it is helpful to use as much sensor information as possible. Followingly, the addition of new sensors to a running recognition system is a task that will be evaluated in detail.

REFERENCES

[1] D. Bannach, "Tools and Methods to Support Opportunistic Human Activity Recognition," Ph.D. dissertation, University of Passau, 2013.

[2] A. Calatroni, D. Roggen, and G. Tröster, "A methodology to use unknown new sensors for activity recognition by leveraging sporadic interactions with primitive sensors and behavioral assumptions," in *Proc. of the Opportunistic Ubiquitous Systems Workshop*, 2010.

[3] M. Stikic, K. V. Laerhoven, and B. Schiele, "Exploring semi-supervised and active learning for activity recognition," in *IEEE ISWC 2008*, 2008, pp. 81–88.

[4] X. Zhu, "Semi-supervised learning literature survey," Computer Sciences, University of Wisconsin-Madison, Tech. Rep. 1530, 2005.

[5] B. Settles, "Active learning literature survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009.

[6] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York NY: Springer New York, 2006.

[7] D. Fisch and B. Sick, "Training of radial basis function classifiers with resilient propagation and variational bayesian inference," in *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*. IEEE, 2009, pp. 838–847.

[8] D. Fisch, M. Jänicke, E. Kalkowski, and B. Sick, "Learning from others: Exchange of classification rules in intelligent distributed systems," *Artificial Intelligence*, vol. 187–188, no. 0, pp. 90 – 114, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370212000410

[9] D. Fisch, M. Jänicke, E. Kalkowski, and B. Sick, "Techniques for knowledge acquisition in dynamically changing environments," *ACM Trans. Auton. Adapt. Syst.*, vol. 7, no. 1, pp. 16:1–16:25, May 2012. [Online]. Available: http://doi.acm.org/10.1145/2168260.2168276

[10] T. Reitmaier and B. Sick, "Active classifier training with the 3DS strategy," in *Computational Intelligence and Data Mining (CIDM), 2011 IEEE Symposium on*. IEEE, 2011, pp. 88–95.

[11] ——, "Let us know your decision: Pool-based active training of a generative classifier with the selection strategy 4DS," *Information Sciences: an International Journal*, vol. 230, pp. 106–131, 2012.

[12] D. Fisch, M. Jänicke, B. Sick, and C. Müller-Schloer, "Quantitative emergence – a refined approach based on divergence measures," in *Self-Adaptive and Self-Organizing Systems (SASO), 2010 4th IEEE International Conference on*, 2010, pp. 94–103.

AIS-based Anomaly Detection in Self-x Systems

Katharina Stahl
Design of Distributed Embedded Systems Research Group
Heinz Nixdorf Institut
Universität Paderborn
Fürstenallee 11
33102 Paderborn
katharina.stahl@uni-paderborn.de

***Abstract*—Self-x behavior introduces novel risks as it may produce undefined system behaviors or states that are not categorically incorrect or malicious. We address this problem by applying anomaly detection for analyzing and evaluating system run-time behavior. Inspired by the Artificial Immune Systems Danger Theory, we propose an anomaly detection mechanism with context-related evaluation. The proposed approach is able to cope with dynamically changing behavior and able to classify behavioral deviations by combining local behavior evaluations with system-wide signals. It seems to be a promising approach to enhance the run-time dependability of a self-x system.**

I. Introduction

Recent requirements on flexibility, intelligence and dynamic behavior in embedded systems may be addressed by means of implementing self-x mechanisms, like self-organization, self-optimization, self-healing etc. Based on resulting behavior complexity, self-x behavior introduces novel risks in terms of dependability by means of autonomous reconfigurations. These autonomous decisions may produce system behaviors or system states which might have not been foreseen or known before. Those undefined behaviors are not categorically incorrect and will not implicitly lead to malicious behaviors in all cases. We argue that it is necessary to integrate means for analyzing and evaluating system run-time behavior. We propose an anomaly detection method for embedded operating systems that is able to cope with autonomously changing behavior and enables a classification of anomalous and unknown behavior at run-time. We use therefor the Dendritic Cells paradigm which is a population-based approach that combines local evaluations with system wide signals (e.g. danger signals) in order to identify malicious system states. We see this approach as a chance to address the problem of evaluating unspecified systems states and, thereby, to enhance the dependability of self-x systems.

A. Artificial Immune Systems

Artificial Immune Systems (AIS) [4],[6] provide a class of principles derived from the human immune system that defends the human body against different, even new and previously unknown, kinds of attacks provoked by intruders like viruses, bacteria, etc. in a robust manner. The potentials of AIS mechanisms can be addressed for computer security and dependability aspects, and especially concerning the purpose of our approach: for anomaly detection.

The *Self-Nonself Discrimination* ([18],[22]) theory, as one main theory, originates from the ability of the human immune system to distinguish between own cells (named *self*) and foreign cells (*nonself*). In the *negative selection algorithm* [4],[6] detectors are generated to match *nonself* in order to identify anomalies. The efficiency of this algorithm relies on accuracy of defining normal behavior (*"self"*) and is governed by the coverage of *nonself* detectors in terms of false alarms. The first approach which used the negative selection algorithm was proposed by Forrest et al. in [10] for virus identification.

The *negative selection algorithm* and the corresponding *positive selection algorithm* do not consider adaptivity and reconfigurability of self-x systems and fail if *self* changes. The *Self-Nonself* theory ignores that nonself does not implicitly need to be dangerous as well as unknown and novel states do not necessarily need to be suspicious. Matzinger proposed the *Danger Theory* in [21] to overcome the problem of only distinguishing between self and nonself. Instead, Danger Theory (with its associated Dendritic Cell Algorithm [11], [12]) considers the context of the system state by means of signals indicating the system's health or damage. These signals trigger the immune response and therefore offers new potentials to anomaly detection in systems that implement reconfiguration and adaptivity.

B. System Background

For the implementation and evaluation, we use ORCOS (Organic Reconfigurable Operating System, see [7]) that was designed by our research group to be a highly customizable and (re-) configurable operating system for embedded systems. ORCOS provides system calls as the only interface for communication between the applications and the operating system due to security aspects. We exploit this interface for our anomaly detection as we define the behavior of a task on the basis of system call sequences.

For implementing self-x capabilities within ORCOS, we extended the ORCOS architecture inspired by the Observer-Controller Architecture [23] from the Organic Computing Initiative [16] allowing us to integrate the famous MAPE cycle [15] (including **M**onitor, **A**nalyzer, **P**lan and **E**xecute). ORCOS contains components for monitoring, analyzing and controlling the operating system.

II. Anomaly Detection in Artificial Immune Systems

Anomaly detection techniques are often applied in the context of intrusion detection systems (IDS) [2] in order to identify unknown system intrusions. Many different concepts have been adopted for anomaly detection apart from artificial immune systems coming from domains like information theory, machine learning, statistics, etc. A survey on the different anomaly detection techniques can be found in [3], [19] and [26]. Anomaly detection approaches for evaluating process behavior have been developed based on the self-nonself discrimination by Forrest et al. [10], [9], [14]. In further work, Forrest et al. examine the number of system calls required for an adequate normal behavior representation in [8]. Danger Theory has actually attracted great attention for anomaly detection techniques [1], [13], [20], [25]. Danger Theory-based approaches applied on system calls are presented in [17]. Twycross et al. show approaches for (process) anomaly detection by means of Danger Theory and Dendritic Cells in [13], [25], [20], [24]. An overview on the recent advances in artificial immune system is given by Dasgupta et al. in [5]. However, up to now, we could not identify any investigations for anomaly detection that operate on autonomously changing behavior.

III. Danger Theory-based Anomaly Detection

Dendritic Cells (DC) with its population-based approach build up the core of our approach. DCs are distributed within the system in such a way that each DC is applied to examine the behavior of a dedicated system component, respectively a task's behavior in terms of e.g. sequences of system calls. Then, the overall system behavior is constructed by assembling and connecting the local behaviors provided by the DCs. The local behavior knowledge is stored in the *shared DC Knowledge Manager* containing the system entire knowledge base. The analyzer is composed of the distributed DCs residing in the system. The resulting architecture for our anomaly detection framework is illustrated in Figure 1.

According to the Danger Theory, four different system-wide signals are provided:

- **Safe Signal:** The emphsafe signal is indicating health of the system and is present if the local behavior of the remaining system components is within the *normal behavior range*.
- **Danger Signal:** The danger signal indicates a potential danger or damage within the system if behavior is identified that does not match *normal behavior*.
- **PAMP *(Pathogen-Associated-Molecular-Pattern)* Signal:** PAMP is an indicator for a known existing system danger.
- **Inflammation Signal:** The inflammation signal is an overall system-wide alarm signal used as an identification for a system reconfiguration.

To detect anomalies in the system behavior, each DC will process the following tasks during its lifecycle:

Step 1. Collect behavioral data and generate representation of actual component behavior

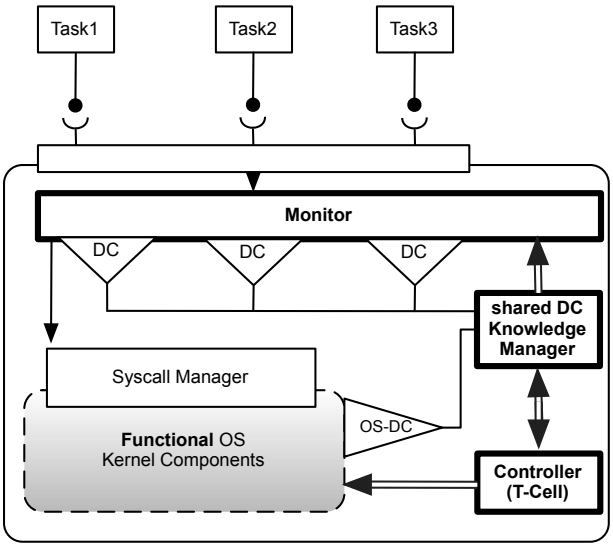

Fig. 1. Introducing Dendritic Cells into the ORCOS operating system

Step 2. Evaluate the behavior (e.g. by (simple) pattern matching)

Step 3. Verify its evaluation with the systems input signals

Step 4. Produce an output signal (enhance the system-wide signals) based on evaluation outcome

The lifecycle of Dendritic Cells is determined by three states: *immature*, *mature* and *semi-mature* state. The initial state of the DC is defined as *immature*. A DC is in *immature* state while proceeding *Step 1* and *Step 2*. From this *immature* state, the DC can migrate either into the *mature* or the *semi-mature* state. The state migration of a DC is depending on its local evaluation outcome.

1) Evaluation Process: In (*Step 1*) the DC it processes data sampling. In *Step 2*, the DC pre-evaluates the monitored local system behavior based on self-nonself discrimination. We are combining positive selection with negative selection in order to efficiently evaluate the local system behavior observed by a DC. The actual behavior is matched against the knowledge base of *normal behavior* by a (simple) pattern matching mechanism. The system-wide signals are used as input signals for evaluation (in *Step 3*) and as output signals (in *Step 4*).

The evaluation procedure defines the following rules:

- If the local behavior matches *normal behavior* (by means of positive selection) and the value of the *danger signal* is within an acceptable range, the DC enhances the *safe signal* and migrates into the *semi-mature state*
- If the local behavior does not match *normal behavior*, the behavior will be classified as an anomaly. To make the anomaly evaluation more precise, the pattern is matched against the *PAMP*-patterns containing all already identified dangerous states (negative selection).
  - If this matching is positive, a threat exists in the system. The *PAMP*-signal will be enhanced.
  - If the behavior pattern does not match the *PAMP*-pattern in the knowledge base, the evaluation process

has to take the value of the *danger signal* into account.

- If a behavioral pattern either matches a *PAMP*-pattern or identifies an anomaly, the DC migrates to the *mature state* and thereby enhances the according output signal.

The evaluation becomes more difficult in the case of a conflictive evaluation compared with the input signals. This happens when the behavioral pattern matches *normal behavior* and the value of the *danger signal* exceeds a defined threshold value. It is also possible to identify an anomaly with a low *danger signal* value and a significant value of the *safe signal*. In such cases, decision making can only be supported by a sophisticated definition of acceptable thresholds.

All evaluation outcomes with their according behavior representing patterns will be stored in the system knowledge base in the *shared DC Knowledge Manager*. This includes (1) the *behavior representation patterns* of all DCs, (2) the *system-wide signals* that are required by the DCs local evaluations and (3) the *PAMP*-patterns of already known threats. The *behavior representation pattern* depends on the requirements of the applied analyzing algorithm that defines what parameters are monitored and need to be collected to establish system behavior data.

2) Reconfiguration: When a system reconfiguration is performed, the actual *normal behavior* becomes obsolete. The danger theory provides the *Inflammation Signal* as a general alarm signal that is used to display a system (re-)configuration. By receiving this signal the DCs are instructed to build up a new system behavior knowledge base within a short time period in order to enable the *shared DC Knowledge Manager* to generate a *normal behavior profile* for this novel configuration. Within this time period, the impact of system-wide signals shall be disabled.

IV. Conclusion

In this paper we address the problem of run-time dependability in embedded self-x systems by applying anomaly detection methods. Anomaly detection in self-x system is a challenging issue. We exploit the Dendritic Cell mechanism offered by the Danger Theory for our approach. This mechanism allows local evaluations of DCs to be coordinated and checked against behavior evaluations of other DCs. The coordination is realized through indirect communication based on system-wide signals and allows context-related evaluations. Reconfiguration is managed by a system-wide signal that is reserved as a general alarm signal. To ensure the performance of anomaly detection, this general alarm signal instructs the distributed DC to adjust their normal behavior profiles immediately. The population-based Dendritic Cell paradigm offers a promising approach for anomaly detection dealing with dynamic behavior in autonomously reconfiguring systems. However, as this work is being in progress, further research is required to answer a couple of still open issues.

References

[1] Uwe Aickelin and Steve Cayzer. The danger theory and its application to artificial immune systems. *CoRR*, 2008.

[2] Stefan Axelsson. Intrusion detection systems: A survey and taxonomy. Technical report, Dept. of Computer Engineering, Chamers University of Technology, 2000.

[3] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 2009.

[4] Dipankar Dasgupta and Fernando Nino. *Immunological Computation: Theory and Applications*. Auerbach Publications, 2008.

[5] Dipankar Dasgupta, Senhua Yu, and Fernando Nino. Recent advances in artificial immune systems: Models and applications. *Applied Soft Computing*, 2011. The Impact of Soft Computing for the Progress of Artificial Intelligence.

[6] L.N. de Castro and J. Timmis. *Artificial Immune Systems: A New Computational Approach*. Springer-Verlag, 2002.

[7] University of Paderborn FG Rammig. https://orcos.cs.uni-paderborn.de/doxygen/html/. ORCOS - Organic Reconfigurable Operating System.

[8] Stephanie Forrest, Steven Hofmeyr, and Anil Somayaji. The evolution of system-call monitoring. In *Proceedings of the 2008 Annual Computer Security Applications Conference*, ACSAC '08. IEEE Computer Society, 2008.

[9] Stephanie Forrest, Steven A. Hofmeyr, Anil Somayaji, and Thomas A. Longstaff. A sense of self for unix processes. In *In Proceedings of the 1996 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 1996.

[10] Stephanie Forrest, Alan S. Perelson, Lawrence Allen, and Rajesh Cherukuri. Self-nonself discrimination in a computer. In *In Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*. IEEE Computer Society Press, 1994.

[11] Julie Greensmith and Uwe Aickelin. The dendritic cell algorithm. Technical report, 2007.

[12] Julie Greensmith, Uwe Aickelin, and Steve Cayzer. Detecting danger: The dendritic cell algorithm. *Computing Research Repository*, 2010.

[13] Julie Greensmith, Jamie Twycross, and Uwe Aickelin. Dendritic cells for anomaly detection. *Computing Research Repository*, 2010.

[14] Steven A. Hofmeyr, Stephanie Forrest, and Anil Somayaji. Intrusion detection using sequences of system calls. *Journal of Computer Security*, 1998.

[15] IBM Corp. *An architectural blueprint for autonomic computing*. IBM Corp., USA, 2004.

[16] DFG SPP 1183 Organic Computing Initiative. http://www.organic-computing.de/spp.

[17] Anjum Iqbal and Mohd Aizaini Maarof. World academy of science, engineering and technology 3 2005 danger theory and intelligent data processing.

[18] Charles A. Janeway, Paul Travers, Mark Walport, and Mark Shlomchik. *Immunobiology: The Immune System in Health and Disease, 6th Edition*. New York and London: Garland Science, 2005.

[19] V. Jyothsna, V. V. Rama Prasad, and K. Munivara Prasad. Article: A review of anomaly based intrusion detection systems. *International Journal of Computer Applications*, 2011. Published by Foundation of Computer Science, New York, USA.

[20] Jungwon Kim, Julie Greensmith, Jamie Twycross, and Uwe Aickelin. Malicious code execution detection and response immune system inspired by the danger theory. In *Proceedings of the Adaptive and Resilient Computing Security Workshop (ARCS-05)*, 2005.

[21] P. Matzinger. Tolerance, danger, and the extended family. *Annual review of immunology*, 1994.

[22] Kenneth M. Murphy, Paul Travers, and Mark Walport. *Janeway's Immunobiology (Immunobiology: The Immune System (Janeway))*. Garland Science, 2007.

[23] Urban Richter, Moez Mnif, Jürgen Branke, Christian Müller-Schloer, and Hartmut Schmeck. Towards a generic observer/controller architecture for organic computing. In *INFORMATIK 2006 ? Informatik für Menschen!*, LNI. Bonner Köllen Verlag, 2006.

[24] Dr Jamie Twycross, Dr Uwe Aickelin, Nottingham Ng Bb, Nottingham Ng Bb, Jamie Twycross, and Uwe Aickelin. An immune-inspired approach to anomaly detection. In *Handbook of Research on Information Assurance and Security. Idea Publishing Group, accepted for publication May 2007. 230*, 2007.

[25] Jamie Twycross, Uwe Aickelin, and Amanda M. Whitbrook. Detecting anomalous process behaviour using second generation artificial immune systems. *International Journal of Unconventional Computing*, 2010.

[26] Yingbing Yu. A survey of anomaly intrusion detection techniques. *J. Comput. Sci. Coll.*, 2012.

Active Learning of Generative and Discriminative Classifiers for Organic Computing

Tobias Reitmaier
Intelligent Embedded Systems Lab, University of Kassel, Germany
Email: tobias.reitmaier@uni-kassel.de

***Abstract*—Organic Computing is based on the insight that human beings are surrounded by systems with massive numbers of processing elements, sensors and actuators. These systems often try to solve, maybe together, a classification problem, e.g. sensor-based activity recognition with the help of mobile devices. For solving this classification problem a classifier must be constructed from unlabeled sample data (sensor measurements) that could mostly be labeled by human domain experts, but only at great expense. Therefore it is neccessary to use a training technique that keeps the costs for constructing a classifier as low as possible without allowing for worse classification results. The category of training techniques to solve such problems can be found in the field of active learning. The goal of the thesis is to develop new active learning strategies that combine information taken from discriminative (support vector machines) and generative (probabilistic) classifiers in order to reduce the number of labeled samples. At the same time the uncertainty of the labeling decision (human domain experts may be erroneous in their decision) and the uncertainty associated with the parameterization of a classifier which is trained with a limited number of training samples should be explicitly modeled and taken into account by the active learning technique.**

***Index Terms*—active learning, generative modeling, discriminative classification, uncertainty modeling**

I. Introduction

Organic Computing (OC) has emerged recently as a challenging vision for future information processing systems. It is based on the insight that we will soon be surrounded by systems with massive numbers of processing elements, sensors, and actuators. Due to the complexity many of those systems will be infeasible to monitor and to control them entirely from external observations. Instead they must monitor, control, and adapt themselves. In order to achieve these goals, these intelligent technical systems must act more autonomously and they must exhibit life-like (organic) properties. Hence, an OC system is a technical system that adapts dynamically to the current conditions of its environment. It is self-organizing, self-configuring, self-healing, self-protecting, self-explaining, and situation-aware [1].

For the interaction of an OC system with humans or other OC systems in dynamic environments the increase of their degree of autonomy in learning is a key to success. In order to obtain their self-x properties, an OC system often solve a classification problem, for example sensor-based activity recognition. Here, sensor networks with novel data mining and machine learning techniques are needed to model a wide range of human activities such as sitting, jumping and running. The sensor data is provided by mobile devices (e.g. smart phones) with enough computing power. For solving this classification problem a classifier must be constructed from unlabeled sample data that could basically be labeled by human domain experts, but only at great expense. Therefore it is neccessary to use a training technique that keeps the costs for constructing a classifier as low as possible without allowing for worse classification results. The class of training techniques to solve such problems can be found in the field of *active learning* [2], [3].

Assuming that the overall set of unlabeled samples (patterns, observations) is available at start of the active training process, techniques from the field of *pool-based active learning* (PAL) [4] can be applied. These techniques build classifiers in a self-organized way: Starting with a set of unlabeled samples, the PAL algorithm actively selects promising (i.e., *informative*) samples that are then labeled by an oracle (e.g. human domain expert). The goal is to keep the number of expert queries as low as possible. The key to success of these PAL techniques is, therefore, the definition of appropriate *selection strategies*.

Active learning can be used in combination with all kinds of learning models. Generally, one can distinguish between *generative* and *discriminative* modeling. Generative models contrast with discriminative models, in that a generative model is a full probabilistic model of all variables, whereas a discriminative model provides a model only for the target variable(s) conditional on the observed variables. Thus, generative models can be used to "generate" new values, are more flexible and expert knowledge can easily be integrated as prior distribution. Discriminative models do not allow to generate samples and often behave as "black boxes", but they yield superior performance for tasks such as classification and regression. This suggests the conclusion to combine the advantages of both worlds within an active learning process.

The three main objectives of the thesis are:

1) Better exploitation of generative information: The most existing state-of-the-art active learning strategies have general disadvantages and make unrealistic assumptions. For example they do not take the initial learning phase and the data distribution during the active learning process into account, and are user-unfriendly, since many parameters have to be set.
2) Combination of generative and discriminative knowledge: Even if the ultimate goal is to build a classifier with good classification performance (a discriminative

classifier), it will be advantageous to use and combine information from a discriminative and a generative classifier to improve the active learning process. It is expected that the number of labeled samples can be reduced compared to existing selection strategies.

3) Consideration of *parameterization* and *labeling uncertainty*: Techniques should be developed that consider these two kinds of uncertainty because they are neclected in existing approaches. The former is justified by the limited training data set, so that the parameters of the classifier cannot be determined with high precision. The latter results from the fact that experts may make errors when they label samples and, thus, several and possibly conflicting expert statements must be fused and appropriately considered.

The remainder of the paper is organized as follows: Section II gives a short overview of the most related work. The main objectives of the thesis are explained in Section III in more detail. Section IV gives the first results and future work. Finally, Section V summarizes the major findings and closes with a short outlook.

II. Related work

In the field of active learning, *membership query learning* (MQL) [5], *stream-based active learning* (SAL) [6] and *pool-based active learning* (PAL) [4] are the most important learning paradigms. MQL and SAL will not be considered here; the former because it may generate artificial samples that cannot be understood or labeled by human experts [2], the latter as it focuses on sample streams. In PAL a number of selection strategies with different properties exists. The most related work can be distinguish in three main categories: uncertainty sampling, density weighting, and diversity sampling.

A rather simple, yet frequently used category of strategies is *uncertainty sampling* [4] which is inspired by the idea of query-by-committee (QBC) [7]. These strategies actively select the sample, for which the current considered classifier (or, according to the original idea of QBC, a committee of classifiers) is most uncertain concerning its class assignment. Strategies belonging to this category are *closest sampling*, *query by bagging*, and *query by boosting*.

Despite the fact that uncertainty sampling has successfully been applied to various problems, a pure uncertainty sampling approach has certain drawbacks, e.g. the chosen samples are potentially erroneous to label and may be outliers. This motivates the category of *density weighting* that considers the distribution of the samples in the input space. Examples for strategies are *density-weighted uncertainty sampling* (DWUS), *dual strategy for active learning* (DUAL), and *prototype based active learning* (PBAC). The strategy presented in [8], DWUS chooses samples that are close to the decision boundary and, at the same time, have a high likelihood (value of the density function) [8]. DUAL, which is introduced in [9], improves DWUS in the following way that samples are selected depending on a (dynamically weighted) convex combination of density measure and uncertainty measure. PBAC [10], a similar approach, selects samples in a self-controlled exploration/exploitation manner.

The selection of only one sample per iteration yields the highest information gain with respect to the number of labeled samples [11], but – depending on the application – it might be more convenient for a human domain expert to label a query set S of samples (with $S > 1$) in each iteration. In such situations the strategies discussed so far tend to select a set of samples that are quite similar to each other, only if the "best" samples according to the strategy are selected. This observation motivates the last category, the *diversity sampling* methods. Basically, this category consists of strategies that combine a measure that ensures that the samples in S are multifaceted with at least one of the other measures. Strategies that belong to this category are *angular diversity sampling* (ADS), a strategy for SVM [12], and *information theoretic diversity sampling* (ITDS) [13], a strategy which is quite similar.

In addition, an overview of current research in the field of active learning can be found in [2], [3].

III. Main objectives

The thesis aims to train a discriminative classifier on a combination of generative and discriminative information sources actively. The main objectives that are needed to achieve this are described in this section in more detail.

A. Better exploitation of generative information

The first aspect focuses on a better exploitation of the properties of the generative classifier. The used generative classifier, called CMM, is based on mixture density model. For more information see [14]. CMM is trained in two steps: A first unsupervised (offline) step and a second supervised (online) step. The second step is repeated in each active learning cycle, since the number of labeled samples increases. With CMM the *responsibility* of a certain model component for the "generation" of a given input sample can be computed. Using that information PAL processes can be avoided where components are neglected or not appropriately considered. This approach offers two important advantages that are neglected by the most state-of-the-art strategies:

1) Many existing PAL techniques do not consider an initial learning phase and start with a given set of labeled samples. The reason is that the choice of an initial query set is quite difficult. By choosing initial query sets that reflect the effective number of samples "generated" by each model component this problem can be solved.
2) Assuming that in the real world all samples "generated" by a specific model component belong to the same class each component can be attributed to a class. Thus, by considering the responsibilities in a selection strategy, it is possible to choose query sets that implicitly consider the (unknown) class distribution of the underlying data set. This solves a key problem of active learning.

B. Combination of generative and discriminative knowledge

In principle, classifiers must have a good classification performance, but there may be other requirements as well. Apart from discriminative classifiers that are "only" expected to have an optimal classification performance on new data (generalization), there are, for example, generative classifiers that additionally aim at modeling the processes from which the observed data originate. Usually, that property comes at the cost of a reduced classification performance but there are many other possible advantages. Support vector machines are an example for the former type, classifiers based on probabilistic (mixture) models an example for the latter. Properties of both will be exploited with the aim to build a discriminative classifier, but with the support of a generative classifier.

Support vector machines (SVM) [15] are based on the principle of structural risk minimization, and they typically have excellent generalization properties. SVM can be trained with optimization techniques such as sequential minimal optimization (SMO) [15]. Most work on active learning deals with SVM (see, e.g., [12], [3]).

In the case of CMM, so-called shared-component models and separate-component models can be distinguished. In the former case, all the component densities are shared between the classes while in the latter case separate mixtures are used to represent each class. CMM based on shared-component models offer the advantage that the mixture density model can partly be trained in an unsupervised way, e.g., with expectation maximization (EM) or variational Bayesian approaches (VI). Class labels are only needed for the assignment of components to classes, i.e., the conditional probabilities for classes given certain components. CMM based on separate-components models are likely to yield a better classification performance.

Research in the field of combining the complementary advantages of discriminative and generative classifiers mainly focuses on techniques that combine these two properties in one paradigm, e.g. relevance vector machines [16]. The thesis differs from the mentioned one that not a training technique should be developed that leads to one classifer that mixes the properties of the two worlds, but a training technique that exploits information from both models in order to construct an SVM in a very efficient way.

How can the generative information, provided from a CMM, be used to train an SVM very efficiently? Three different approaches are investigated:

1) The active selection strategy considers the generative information by itself. This can be done, if a CMM and an SVM are actively trained as ensemble. The selection strategy chooses, e.g., samples that lie in high-density regions (can be seen as prototypes) or according to the data distribution (both kinds of information can be provided by the CMM) and samples for which the SVM is most uncertain concerning its class assignments. The selection strategy can combine these different measures with help of a linear combination.
2) A separate-component model is trained actively because it provides better classification performance. The problem is that for training of such models a completely labeled training set is required that takes the data distribution of the whole data set into account. To overcome this issue a shared-component model is trained actively with an appropriate selection strategy and in each learning cycle the yet unlabeled samples are labeled by the shared-component model in a semi-supervised manner. This provides in each active learning cycle a training set for the separate-component model.
3) The generative information provided by the CMM is mapped into the SVM kernel function (used to map the data into a high dimensional feature space, where the data can be separated [15]). This can be done with help of a new kernel function that uses an appropriate similarity measure that is based on the underlying mixture model of the CMM. To measure the similarity of two samples, a weighted linear combination of Mahalanobis distances is used, because normally a Mahalanobis distance can be calculated only with respect to one Gaussian normally. Doing so, the SVM has the same structure information as the CMM and therefore the active training of the SVM is more efficiently.

C. Consideration of parameterization and labeling uncertainty

The following two kinds of uncertainty are neglected in related work of active learning. The goal is to develop techniques that consider these uncertainties and therefore perform better than techniques that neglect these.

Parameterization uncertainty is the consequence of a limited number of available training samples. In the case of CMM, the exact values of the parameters of the mixture model can not be determined with certainty. In the case of the SVM, the set of support vectors and their weights are not exact determinable.

In the case of CMM, the parameterization uncertainty will be considered by using the explicit uncertainty measured from the training algorithm. Basically, the idea is to consider "worst cases" in the case of great uncertainty. That is, the selection strategies must be modified, e.g., in order to consider smaller/larger areas in the input space (if density information is used for selection) or to consider a smaller/larger number of samples for specific components over several iterations of the PAL process (if mixing coefficients or responsibilities are used for selection).

In the case of SVM, parameters are influenced by the cost parameter C and the kernel parameter γ (kernel width) in the case of Gaussian kernels (for details, see [15]). Slightly different data sets, for instance, may lead to completely different sets of support vectors. Thus, it might be problematic to start with a (randomly selected) small set of labeled samples and to select additional samples for labeling depending on the distance to the decision boundary which is modeled by these samples. To overcome this problem, the PAL process is started with a broad soft margin (low value for C, for instance) and during the PAL process this margin is narrowed.

Labeling uncertainty is the consequence of possible errors made by human domain experts who label actively selected samples. It is obvious that more errors will occur for samples close to the (initially unknown) decision boundary. These samples, however, are needed for a fine-tuning of the decision boundary. On the assumption that experts may make errors, as a consequence, multiple labeling of samples is allowed (cf. the concept of "sampling with replacement"). It is important to mention, however, that an expert statement may be gradual and be either real or fictive. The latter will be used to start the labeling process for a sample with a non-informative initial prior (uniform distribution) that has a weak influence on the final result. For modeling the expert statements a dirchlet or beta distribution is used. At any time during the PAL process, the labeling process provides a gradual label for each sample and a measure for the uncertainty concerning this label. In a next step, the training algorithms for SVM and CMM will be modified to consider gradual labels. Finally, the uncertainty measures are considered in our selection strategies to select samples with higher probability if its label is more uncertain.

IV. First results and future work

To meet the challenges that are addressed in sections III-A and III-B-1 most of related state-of-the-art strategies are implemented and evaluated. In addition, two new learning strategies that exploit generative information for sample selection are already published:

The first selection strategy is called *distance-density-diversity sampling* (3DS) [17]. 3DS considers the distance of samples to the (estimated) decision boundary in order to define this decision boundary precisely, the density of samples in order to avoid the selection of outliers, and the diversity of samples in the query set that are chosen for labeling. 3DS combines these measures in a self-adaptive manner, only the weighting factor for the diversity measure should be set by the user. If the size of the query set is one, 3DS can be termed to be parameter-free.

In [18], an extention to 3DS, called 4DS, is presented, that also recognizes the *distribution* of samples: Assuming that the components of the mixture model should uniquely be assigned to classes, the unknown class distribution of samples should be considered by using the responsibility of each model component for a sample as a new decision criterion. Doing so, 4DS solves a key problem of active learning: The class distribution of the samples chosen for labeling actually approximates the unknown "true" class distribution of the overall data set quite well. In order to combine the four mentioned criteria, 4DS uses a weighted linear combination, in which the weighting factors are found in a self-optimizing way. Further, it is shown that responsibility information derived from generative models can successfully be employed to improve the active training process of discriminative classifers.

A publication on the subject, which addresses section III-B-3, is currently under preparation.

V. Conclusion and Outlook

In this paper the active training of a discriminative classifier (SVM) on a combination of generative and discriminative information sources is addressed. In addition, this active training process should be efficient in terms of labeling costs and user-friendlyness. From OC perspective, this work can pave the way to build OC systems that efficiently and effectively learn with minimal feedback from their (dynamic) environment by using knowledge about their own deficiencies (a kind of self-assessment or self-awareness).

References

[1] K. Bellman and P. Hofmann and C. Müller-Schloer and H. Schmeck and R. P. Würtz, *Executive Summary – Organic Computing – Controlled Emergence*, Schloss Dagstuhl, Germany, 2006.

[2] B. Settles, Active Learning Literature Survey, Computer Sciences Technical Report 1648, University of Wisconsin, Department of Computer Science, 2009.

[3] J. Jun, I. Horace, Active learning with SVM, in: J. Ramn, R. Dopico, J. Dorado, A. Pazos (Eds.), Encyclopedia of Artificial Intelligence, vol. 3, IGI Global, Hershey, PA, 2009, pp. 1 – 7.

[4] D. D. Lewis, W. A. Gale, A sequential algorithm for training text classifiers, in: Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 94), Dublin, Ireland, 1994, pp. 3 – 12.

[5] D. Angluin, Queries and concept learning, Machine Learning 2 (4) (1988) 319342.

[6] L. Atlas, D. Cohn, R. Ladner, M. A. El-Sharkawi, R. J. Marks II, Training connectionist networks with queries and selective sampling, Advances in Neural Information Processing Systems, vol. 2, Morgan Kaufmann, Denver, CO, 1990, pp. 566 – 573.

[7] H. S. Seung, M. Opper, H. Sompolinsky, Query by committee, in: Proceedings of the Fifth Annual ACM Conference on Computational Learning Theory (COLT 92), Pittsburgh, PA, 1992, pp. 287 – 294.

[8] H. T. Nguyen, A. Smeulders, Active learning using pre-clustering, in: Proceedings of the Twenty-First International Conference on Machine learning (ICML 04), Banff, AB, 2004, pp. 79.

[9] P. Donmez, J. G. Carbonell, P. N. Bennett, Dual strategy active learning, in: Proceedings of the 18th European Conference on Machine Learning (ECML 07), Warsaw, Poland, 2007, pp. 116 – 127.

[10] N. Cebron, M. R. Berthold, Active learning for object classification: from exploration to exploitation, Data Mining and Knowledge Discovery 18 (2) (2009) 283 – 299.

[11] C. A. Thompson, M. E. Califf, R. J. Mooney, Active learning for natural language parsing and information extraction, in: Proceedings of the 16th International Conference of Machine Learning (ICML 99), Bled, Slovenia, 1999, pp. 406 – 414.

[12] K. Brinker, Incorporating diversity in active learning with support vector machines, in: Proceedings of the 20th International Conference on Machine Learning (ICML 03), Washington, DC, 2003, pp. 59 – 66.

[13] C. K. Dagli, S. Rajaram, T. S. Huang, Utilizing information theoretic diversity for SVM active learning, in: Proceedings of the 18th International Conference on Pattern Recognition (ICPR 06), Hong Kong, China, 2006, pp. 506 – 511.

[14] D. Fisch and B. Sick, *Training of radial basis function classifiers with resilient propagation and variational Bayesian inference*, In *International Joint Conference on Neural Networks (IJCNN '09)*, pages 31–38, Nashville.

[15] V. N. Vapnik, An overview of statistical learning theory, *IEEE Transactions on Neural Networks*, 10(5):988 – 999, 1999.

[16] M. E. Tipping, The relevance vector machine, In *Advances in Neural Information Processing Systems (NIPS '00)*, volume 12, pages 652 – 658, Vancouver, BC, 2000.

[17] T. Reitmaier and B. Sick, Active classifier training with the 3DS strategy, In *IEEE Symposium on Computational Intelligence and Data Mining (CIDM '11)*, pages 88 – 95, France, Paris, 2011.

[18] T. Reitmaier and B. Sick, Let us know your decision: Pool-based active training of a generative classifier with the selection strategy 4DS, In *Information Sciences Journal*, volume 230, pages 106 – 131, Elsevier, 2012.

Distributed Management of Cloud Computing Applications

Michael Roth
Universität Augsburg, Germany
E-Mail: michael.roth@informatik.uni-augsburg.de

I. Introduction

Cloud Computing allows customers to pay for servers as needed. Customers can deploy large web applications without an upfront investment in hardware. Most web applications have a high peak load in daytime and fewer visitors at night. Because of the ability to change the number of running servers in a matter of minutes the number of servers can be reduced during times of low traffic, and it can be increased during peak times. By using Cloud Computing developers can also start a small web application which can easily scale up if more users discover the application. For these reasons Cloud Computing has become very popular over the last years.

Large Cloud Computing applications consist of thousands of servers and are too dynamical to manage them manually. Therefore, an automated way to manage these applications is needed. Organic Computing aims to control complex systems by introducing self-x abilities. Organic Computing systems posses the abilities to self-configure, self-optimise, self-heal and self-protect themselves.

We are developing an Organic Computing Middleware [1] which is capable of managing large Cloud Computing applications on its own by using these self-x properties. Our middleware is completely decentralised and therefore possesses no single-point-of-failure. An Organic Manager is the key component of our middleware. The manager uses a MAPE Cycle [2] to observe and control the system. The planing in our middleware is done by an automated planer [3]. We investigate the monitor and analyse phases in this paper.

II. Scenario

In our scenario a company develops a new web application and uses Cloud Computing to run the application. The company rents virtual machines from a Cloud Computing provider. The application requires web servers to process user requests and database servers to store information.

We use Amazon Web Services (AWS) [4] as an example Cloud Computing provider. Other providers have similar services, therefore our research can be used with them as well. Users can rent Virtual Machines (VM) and pay for them by the hour. VMs have different computing capabilities. The more computing power a VM possesses the higher is the price. Currently the prices for one hour range from $0.02 to $4.60 for the AWS Data Centre in North Virginia. Additionally, customers are charged for generated network traffic.

The customers have only limited influence on the VM placement. AWS currently operates 8 data centres, called Regions, all over the world. Each Region is divided in two to four Availability Zones. A customer can choose the Region and the Availability Zone for new instances.

Amazon also offers some additional services to manage the cloud application. Amazon's monitoring service CloudWatch [5] is a centralised monitoring system which can monitor instances in one Region. It is not possible to monitor the whole application over multiple Regions. The CloudWatch server collects information from the instances over SNMP. The user can also define other information sources.

With Amazon's Auto Scaling new VMs can be deployed automatically if an observed metric reaches a predefined threshold. It is also possible to stop instances if an observed metric falls below predefined value. The Auto Scaling service uses the data collected from CloudWatch. It is only possible to define thresholds which trigger actions.

Elastic Load Balancing [6] is a service from Amazon which manages user requests and distributes them to different servers. Users send their requests to the load balancer. The load balancer distributes the requests to the web server pool. Therefore it is possible to change the number of web servers without the user noticing. The Amazon load balancer can only handle traffic for one region.

In our work we use an example web application which is used by customers all over the world and is therefore hosted in all 8 AWS regions. To handle the failure of an entire region the monitoring system must observe all VMs in all regions. It is not necessary to know the exact information for each VM in other regions but the overall state must be known. We use the response time of the web server as indicator for the application's health. Users expect an answer in a reasonable time and are not interested on the CPU load or RAM usage of the instances. If the response time is too high the middleware must determine the cause. A high response time can be caused by the web server or by the database server. If only a single web server is slow, succeeding requests can be routed to other web servers by the load balancer until the slow web server recovers. If more web servers have a high response time the middleware must decide if new instances must be launched. To save money the company wants only to rent the number of servers which are actually required to handle the user requests within a given response time interval. If the workload decreases and the user requests are answered with a very low

response time the middleware can shut down instances to save money.

The middleware must also detected failed instances and decide whether the VMs must be replaced or shut down. Such failed instances can respond fast to user requests with an error page or a partial HTML page. Therefore not only a low response time must be guaranteed. The middleware must also ensure that the web servers work correctly.

III. Monitoring Phase

Amazon Web Services use a centralised monitoring system like most Cloud Computing providers. In such a system all monitoring data is sent to the central monitoring instance. To avoid a singe-point-of-failure the monitoring instance is often replicated which leads to even more traffic. Amazon CloudWatch is not suitable for our scenario since it can not monitor system in different AWS regions.

Our goal is to develop a decentralized monitoring system for a self-managing Cloud Computing Middleware. The monitoring system must collect enough information to observe the status of the web application. With the monitoring information the middleware can decide if the system must be adjusted. Some instances in the web application monitor the system. These instances are chosen by the middleware. We want to spread these observer instances over the entire network. Each observer is responsible for close-by instances but knows also the approximated status of the entire network. If an observer fails or the system changes, the middleware can change the number of observers or relocate the observer to another VM.

The instances are not aware of the network structure. To enable the instances to send monitoring data to all observers, all components form a distributed hash table (DHT). The DHT allows a structured decentralised forwarding of information for all participants in the middleware. More details on the used DHT based information dissemination algorithms can be found at [7].

To allow the observer instances to receive the required information a publish/subscribe protocol is used. We use the DHT network to forward the subscriptions and information. Neighbours in the DHT network are chosen by their network distance. The node information of different nodes is combined on its way to the observer. Therefore each node in the information path has an partial view an the network. Only the observer receives all information and has therefore a more complete view of the network. Because of the different transit times the observer does not possess a consistent view.

IV. Analyse Phase

Because of the decentralized nature of our middleware and to save network bandwidth we want to analyse the information on its way to the monitoring instances and send only aggregated information. We also want to investigate if it is possible to trigger actions before the information reaches the monitoring instance if the aggregated data shows a fatal problem. This can be done by the nodes forwarding the information to the observer. In this case the observer is only responsible for optimizations that cannot be performed by the other instances. The monitoring system must also observe the actions taken from aggregated information and control if these actions guide the system into a valid state.

A. Fuzzy Logic

One interesting way to analyse the monitored data is Fuzzy Logic. In Fuzzy Logic variables have truth values which can adopt a value between 0 and 1. Member Functions describe the truth values of variables for given input values. Rules are used to map the variables to output variables. The truth values of the input variables and the rules are used to calculate the truth values of the output variables. By using the different truth values of the output variables an output value is calculated.

Usually the input and output variables are named to be easily understood, e.g. *low utilization*, *high temperature*, *moderate latency*, *extreme high throughput*, *start few instances*, *start many instances*. The rules are therefore very easy to understand, e.g. IF *high latency* AND *high throughput* THEN *start few instances*. The idea is that these simple rules can be entered by users without knowing the exact values representing good and bad latencies. The definition of the Member Functions must be done by an expert whereas the rules can be generated by users without specific domain knowledge. Because of this simple rule language fuzzy logic is a good match for our organic computing concept.

B. Time Series Analysis

Most web applications have a peak load every day at the same time and a different load on weekends and holidays. With time series analysis and time series forecasting we want to analyse the previous behaviour to predict such peak times. If possible such analysis should be done by each instance. The instances send recommendations to the monitoring system. These recommendations can be calculated on the source instances and are sent to the monitoring instances. The monitoring instances can trigger actions on behalf of these recommendations.

We want to investigate if it is possible for single instances to calculate reliable forecasts. Each instance knows only the local load information. If the VM has recently been started there is not enough information available. If the local knowledge is not sufficient enough the load information is sent to the monitoring instance. We want to determine how many data is needed to calculate a reliable forecast. The data sent to the monitoring system can be aggregated on its way. A forecast can be calculated if the aggregated data is significant so that only the forecast for a group of instances is sent to the monitoring system. By doing so we can save bandwidth.

V. Evaluation

Because of the size of such web applications we are unable to test our scenario in the real world evaluation. We use the ns3 [8] network simulator to generate a simulated Cloud Computing environment. The work of Barroso and Hölzle [9] are used to model a Cloud Computing data centre. Within this

environment we will simulate user interactions. We are looking for real performance information to simulate the behaviour of the VM instances and the visitors. We will induced VM failures and fluctuations in the number of visitors. To judge the efficiency of our middleware we will compare our approach with traditional centralised solutions. The objective of our evaluation is to see if the example web application can be reached with a low response time and at the same time the number of used servers is kept low. To measure the costs we will monitor the used network bandwidth. Also the time required to counteract disturbing influences will be measured.

VI. Summary

We presented a distributed middleware to manage cloud computing application. The middleware uses an organic manager to control the application. The manager uses a MAPE cycle. In this paper we focus on the monitoring and analyse phase.

For information dissemination the instances of the application build a distributed hash table network. Information can be spread throughout the entire network without knowing the network topology or many other active instances by using the DHT network. We use a publish/subscribe protocol over the DHT network to send the node information only to the observer instances. All other nodes collect and forward the information to these observers.

A distributed analyse phase aggregates the information on its way to the observer. If a problem is detected which can be solved with local knowledge actions are taken. The observer must only interact with the system if a bigger problem is detected.

We will evaluate our research with a network simulator. To get accurate results we model the network after current cloud computing centres and use real performance information for modelling the instances behaviour.

References

[1] M. Roth, J. Schmitt, R. Kiefhaber, F. Kluge, and T. Ungerer, "Organic Computing Middleware for Ubiquitous Environments," in *Organic Computing A Paradigm Shift for Complex Systems*, ser. Autonomic Systems, C. Müller-Schloer, H. Schmeck, and T. Ungerer, Eds. Springer Basel, 2011, pp. 339–351.

[2] J. Kephart and D. Chess, "The Vision of Autonomic Computing," *IEEE Computer*, pp. 41–50, jan 2003.

[3] J. Schmitt, M. Roth, R. Kiefhaber, F. Kluge, and T. Ungerer, "Using an Automated Planner to Control an Organic Middleware," in *Self-Adaptive and Self-Organizing Systems (SASO), 2011 Fifth IEEE International Conference on*. IEEE, 2011, pp. 71–78.

[4] Amazon, *Getting started with Amazon Web Services*, 2013 (accessed April 22, 2013). [Online]. Available: https://aws.amazon.com/en/documentation/gettingstarted/

[5] ——, *Amazon CloudWatch Documentation*, 2013 (accessed April 22, 2013). [Online]. Available: https://aws.amazon.com/en/documentation/cloudwatch/

[6] ——, *Amazon Elastic Load Balancing Documentation*, 2013 (accessed April 22, 2013). [Online]. Available: https://aws.amazon.com/en/documentation/elasticloadbalancing/

[7] M. Roth, J. Schmitt, F. Kluge, and T. Ungerer, "Information dissemination in distributed organic computing systems with distributed hash tables," in *Computational Science and Engineering (CSE), 2012 IEEE 15th International Conference on*. IEEE, 2012, pp. 554–561.

[8] N. Baldo, M. Requena, J. Nunez, M. Portoles, J. Nin, P. Dini, and J. Mangues, "Validation of the ns-3 IEEE 802.11 model using the EXTREME testbed," in *Proceedings of SIMUTools Conference, 2010*, March 2010.

[9] L. Barroso and U. Hölzle, "The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines," *Synthesis Lectures on Computer Architecture*, vol. 4, no. 1, pp. 1–108, 2009.

Towards a decentralized intelligent traffic management system

Matthias Sommer
Organic Computing Group
University of Augsburg
Email: matthias.sommer@informatik.uni-augsburg.de

***Abstract*—Growing cities and the increasing number of inhabitants lead to a higher volume of traffic in urban road networks. As space is limited and the extension of existing road infrastructure is expensive, the construction of new roads is not always an option. Therefore, it is necessary to optimise the existing urban road network to reduce the negative effects of traffic, e.g. pollution emission and fuel consumption. Urban road networks are characterised by their great number of signalised intersections. Until now, the optimisation of these signalisations is done by hand through traffic engineers. As urban traffic demands tend to constantly change, it is almost impossible to foresee every situation upfront. Hence, a new approach is needed, that is able to react adaptively at run-time to optimize signalisations of intersections according to the current situation. The Organic Traffic Control (OTC) system offers a decentralised approach with communicating intersections, which are able to adapt their signalisation dynamically at run-time and establish progressive signal systems to optimize traffic flows and the number of stops per vehicle.**

I. Introduction

Urban road networks are characterised by their great number of signalised intersections. Traffic engineers try to use the existing road network efficiently by optimising signal plans followed by an improved coordination of traffic flows in order to reduce negative impacts of traffic like pollution emissions. The problem here is, that even the optimisation of a single signalised intersection is a very difficult task due to its mathematical complexity.
The increasing mobility and rising traffic demands cause serious problems in urban road networks. An additional difficulty arises from the fact that traffic demands in urban road networks are constantly changing, so that the signalisation has to be continuously adapted to new situations. Under the assumption that these changes would appear at frequent intervals it would be possible to handle them with time-dependent switching of predefined signal plans. As there are irregular traffic demands due to events like sport events or the beginning or ending of holidays or even completely blocked roads due to incidents, road works or bad weather conditions, which are difficult or even impossible to be foreseen, it is necessary to shift the signal plan optimisation from design-time to run-time in order to be able to immediately adapt the signalisations of intersections. Therefore, learning intersections are needed that can communicate among each other and adapt their signalisation autonomously to changes in the environment.

A. Related work

Approaches to reduce the negative impacts of traffic include an improved control of traffic lights and the introduction of dynamic traffic guidance systems that take current conditions into account. The *Split, Cycle, and Offset Optimisation Technique* (SCOOT, [1]) is one of the first adaptive network control systems that was successfully applied in the field. It responds automatically to fluctuations in the traffic flow and offers other techniques like the priorisation of busses. The *Sydney Coordinated Adaptive Traffic System* (SCATS, [2]) is a computerised area wide traffic management system started in 1970 and continually being improved. It is able to control traffic intersections and mid-block pedestrian crossings. Further projects dealing with new approaches for the management of traffic are *OPAC* (Optimisation Policies for Adaptive Control, [3]), *BALANCE* (Balancing Adaptive Network Control Method, [4]) and *MOTION* (Method for the Optimisation of Traffic Signals in Online-Controlled Networks, [5]).
Another solution for the former aspect is Organic Traffic Control (OTC) which provides a self-organised and self-adaptive system founding on the principles of Organic Computing (OC, [6]). The design principle behind this architecture is to transfer characteristics like *local responsibility*, *self-organisation*, *robustness*, *adaptivity*, and *capability of learning* to systems of different application domains. The current status of the OTC system is mainly based on dissertations of [7] and [8]. Based on these works, intersection controllers got the ability to self-adapt to changing traffic conditions by adapting their signalisation. Furthermore, the system is able to establish a coordinated operation of nearby intersections to enable so-called green waves. In contrast to the formerly presented approaches, OTC relies on a decentralised structure which eliminates the problem of a single point of failure and the bottleneck between the central traffic management unit and the other components in the system.

B. Working plan

Future work will further improve the OTC system by enhancing the framework with several techniques. An intelligent traffic management system has to provide features like traffic flow prediction, methods for congestion avoidance and congestion detection. These abilities will improve the system to be able to deal with complex road traffic situations. These features will reduce the average travel time through the network as well

as the number of stops. Additionally, public transport will be included in the optimisation process of the signalisation, allowing the system to give priorisation to buses or trams. These goals will be presented in the following.

II. Objectives and Approach

A. State of the art

The OTC system consists of a self-organising observer/controller framework for signal control. As shown in Fig. 1, the OTC architecture extends the intersection controller within an existing road network, the System under Observation and Control (SuOC), by adding several layers on top. Current installations in cities worldwide often rely on so-called fixed-time controllers. These controllers follow pre-defined phases in which different turnings receive the right to switch to green. Additionally, so-called interphases are defined, in which all traffic lights of an intersection are switched to red light to avoid accidents.

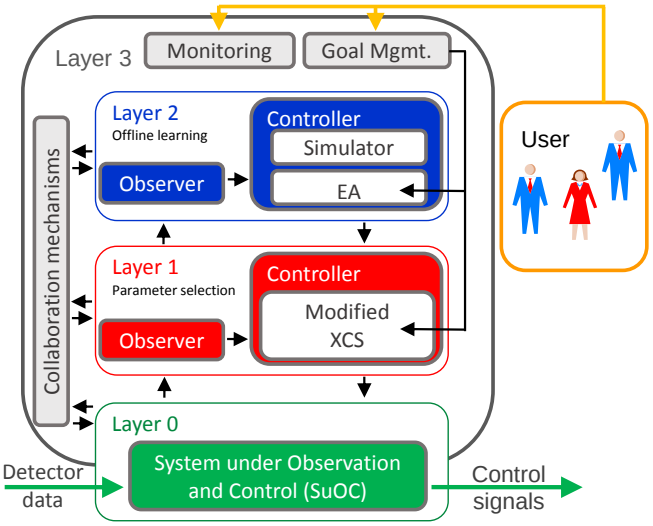

Fig. 1. Organic Traffic Control Architecture

Therefore, the traffic light controller follows a simple time-dependent algorithm. Fig. 2 depicts a simple intersection with 12 turnings and detectors. These detectors are typically implemented as induction loops in the surface and measure traffic flows for every turning.

The recorded traffic flows are passed to the observer on Layer 1. This Layer 1 contains a modified learning classifier system (based on Wilson's XCS [9]) where parameter sets for the signalization, based on the observed traffic flow data, are selected. In case of a new situation (no parameter set is known), the offline learning component on Layer 2, represented by an evolutionary algorithm, creates new classifiers and passes these back to Layer 1, where the new parameter set is applied. Simultaneously, Layer 1 reacts with the best possible action while Layer 2 searches for a new solution. Here, AIMSUN (Advanced Interactive Microscopic Simulator for Urban and Non-Urban Networks [10]) is used to evaluate

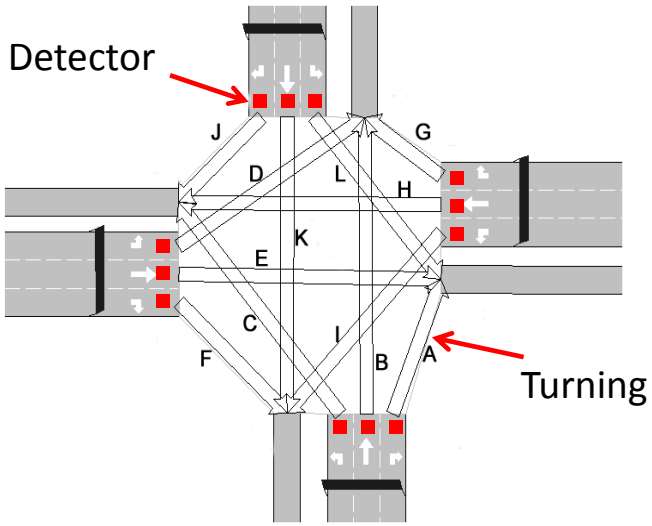

Fig. 2. An exemplary intersection with detectors and turnings

the quality of the created parameter sets. Mechanisms for a decentralized collaboration between intersections enable these to communicate and exchange data. This allows the network to coordinate signalisations of traffic lights in response to changing traffic demands. The system is able to reduce the number of stops per vehicle to reduce pollution emission, fuel consumption and travel time, while maximizing the flow through the road network. As several intersections of an urban road network can be located in close vicinity, their coordinated signalisation is essential for implementing so-called green waves to increase the traffic flow. By identifying the strongest traffic streams, the system is able to minimise the network-wide number of stops by establishing green waves for these streams.

B. Future work

As the traffic management system gets the ability to self-adapt, mechanisms must be introduced for ensuring the correctness of the system's behavior and the applied actions. Therefore the road network evolves to an robust intelligent decentralised traffic management system, which can autonomously adapt within defined borders and reliable offer advantages for traffic participants. By further introducing techniques like traffic forecasting and early congestion detection, the system transforms from a reactive to a proactive traffic management system.

1) Predicting future traffic situations: The existing OTC system [7] is able to react to changes in the traffic flow, but it lacks the ability to proactively establish mechanisms to prevent the traffic network of traffic jams or traffic bottlenecks. Forecasting the upcoming traffic flow may help the system to prevent decreases in traffic flow caused by road constructions or bad weather conditions. As [11] states, artificial neural networks (ANN) were already successfully explored for the prediction of traffic flows up to 15 minutes ahead. This

approach has shown to be able to deal with complex nonlinear predictions [12], which lets ANNs appear as appropriate for the traffic domain. A multitask learning (MTL) model for ANNs will be used as presented in [13] and [14], as MTL may offer improvements to the generalization performance of the ANN by integrating field-specific training information contained by the extra tasks. The most considered task is the so-called main task, while the others are called extra tasks. The introduced prediction techniques will be integrated into the observer on Layer 1, so every intersection controller has a prediction component for each turning. Furthermore the raised data should be integrated in the components responsible for the routing algorithm and signalisation of the traffic flow. As the simulation of new classifiers on Layer 2 needs some time, the situation passed to Layer 2 could be the predicted one, instead of the current one. With a sophisticated prediction, this approach might lead to a faster reaction time and a better matching set of classifiers especially in the startup phase of the system.

2) Dealing with congestions: By an early detection of possible capacity shortages, the system should be able to adapt the routing of vehicles to avoid congestions. But even with a sophisticated forecasting technique, situations which can not be foreseen may exist. Accidents are not predictable and may have severe influences on the traffic situation. An accident may not only lead to a drastic decrease in traffic flow in the particular location, but may also cause a tailback that blocks other routes that are not overloaded [15]. This phenomenon especially occurs in busy areas with a large number of road intersections. The OTC system has therefore to be equipped with two new features. The first is a detection mechanism for congestions which may be achieved by analysing the data passed by detectors and the resulting traffic flow data. Several papers were written about this topic. Approaches include vehicle-to-vehicle communication [16], stationary video cameras [17] or cellular system technology [18]. In the OTC system, data from detectors in the street surface should be incorporated in the congestion detection mechanism. In case of a detected traffic shortage, the second mechanism then adapts the system in the way that the traffic flow takes other paths through the system, in the way that the negative effect of the accident is minimized. This may be achieved by adapting green times of traffic lights and an extended route guidance through variable message signs. The signalising should therefore consider (dynamic) link capacities. The OTC system is already equipped with a decentralised routing component, that is able to determine the best routes to prominent destinations in the network for unaffected traffic flows [7]. This component should now be extended by the described mechanisms. An interesting effect might be observed, when all or most traffic participants follow the alternative route proposal. By choosing the alternative route, this might lead to new traffic jams on the alternative routeing. Therefore, a (distributed) detection of the emergent effect of cascading link failures should be established. In addition, an intelligent algorithm is needed, to distribute the traffic participants adequately on alternative paths through the road network.

3) Public transport: Until now, the OTC system only considers individual traffic. By introducing new vehicle classes for public transport like buses or trams, the consideration of these traffic participants will also be part of the signalisation optimisation process. Therefore, the system will be able to give priority to public transport by extending corresponding green times or shortening conflicting phases to allow unaffected passing by the vehicle. This may lead to decreased waiting times and number of stops for public transport, with possible negative effects for other traffic participants.

4) Optimising phase sequences: A further improvement should by achievable by incorporating all basic signalisation parameters into the optimisation process. Currently, only the phase durations, the cycle time - and in the coordinated case - the offset of signalised intersection have been considered. By establishing a predefined set of allowed phase transitions, the phase sequence can be incorporated as additional parameter. This leads to decreased waiting times as costly phase transitions might be skipped.

III. Conclusion

The realisation of the presented ideas enhances the existing OTC system to convert it from a reactive to a robust proactive traffic management system which is able to predict upcoming traffic shortages and drops in traffic flow and, in addition, has the ability to react autonomously and to adapt the traffic flow to solve the presented problems. By including public transport into the optimisation process of the network, the system is able to handle these prioritised. Therefore, the OTC system gets the ability to detect traffic anomalies and is assigned with the feature of self-organised resilience.

References

[1] D. I. Robertson and R. D. Bretherton, "Optimizing networks of traffic signals in real time – the SCOOT method," *IEEE Transactions on Vehicular Technology*, vol. 40, no. 1, pp. 11–15, 1991.

[2] A. G. Sims and K. W. Dobinson, "The Sydney coordinated adaptive traffic (SCAT) system – Philosophy and benefits," *IEEE Transactions on Vehicular Technology*, vol. 29, no. 2, pp. 130–137, 1980.

[3] N. H. Gartner, "OPAC Strategy for demand-responsive decentralized traffic signal control," in *Control, Computers, Communications in Transportation*, J.-P. Perrin, Ed., 1989.

[4] B. Friedrich, "Steuerung von Lichtsignalanlagen, BALANCE - ein neuer Ansatz," *Straßenverkehrstechnik, Kirschbaum Verlag, Bonn, DE*, vol. 7, 2000.

[5] J. Mück, "Neue Schätz- und Optimierungsverfahren für Adaptive Netzsteuerungen," *Straßenverkehrstechnik*, vol. 52, pp. 761–773, 2008.

[6] C. Müller-Schloer, C. von der Malsburg, and R. P. Würtz, "Organic Computing," *Informatik Spektrum*, vol. 27, no. 4, pp. 332–336, 2004.

[7] H. Prothmann, *Organic Traffic Control*. KIT Scientific Publishing, 2011.

[8] S. Tomforde, *Runtime adaptation of technical systems: An architectural framework for self-configuration and self-improvement at runtime*. Südwestdeutscher Verlag für Hochschulschriften, 2012, iSBN: 978-3838131337.

[9] S. W. Wilson, "Classifier fitness based on accuracy," *Evolutionary Computation*, vol. 3, no. 2, pp. 149–175, 1995.

[10] J. Barceló, E. Codina, J. Casas, J. Ferrer, and D. García, "Microscopic traffic simulation: A tool for the design, analysis and evaluation of intelligent transport systems," *Journal of Intelligent and Robotic Systems*, vol. 41, no. 2–3, pp. 173–203, 2005.

[11] R. Chrobok, O. Kaumann, J. Wahle, and M. Schreckenberg, "Different methods of traffic forecast based on real data," *European Journal of Operational Research*, vol. 155, no. 3, pp. 558–568, June 2004. [Online]. Available: http://ideas.repec.org/a/eee/ejores/v155y2004i3p558-568.html

[12] E. Bolshinsky and R. Freidman, "Traffic flow forecast survey," Technion - Israel Institute of Technology, Tech. Rep., 2012. [Online]. Available: http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/2012/CS/CS-2012-06.pdf

[13] F. Jin and S. Sun, "Neural network multitask learning for traffic flow forecasting," in *IJCNN*, 2008, pp. 1897–1901.

[14] G. Orr and K.-R. Müller, *LNCS 1524*. Springer, 1998.

[15] B. Immers, J. Stada, I. Yperman, and A. Bleukx, "Towards robust road network structures," in *Slovak Journal Of Civil Engineering*, 2004.

[16] R. Bauza, J. Gozalvez, and J. Sanchez-Soriano, "Road traffic congestion detection through cooperative vehicle-to-vehicle communications," in *Proceedings of the 2010 IEEE 35th Conference on Local Computer Networks*, ser. LCN '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 606–612. [Online]. Available: http://dx.doi.org/10.1109/LCN.2010.5735780

[17] V. Jain, A. Sharma, and L. Subramanian, "Road traffic congestion in the developing world," in *Proceedings of the 2nd ACM Symposium on Computing for Development*, ser. ACM DEV '12. New York, NY, USA: ACM, 2012, pp. 11:1–11:10. [Online]. Available: http://doi.acm.org/10.1145/2160601.2160616

[18] R. P. W. Pattara-atikom and R. Luckana, "Estimating road traffic congestion using cell dwell time with simple threshold and fuzzy logic techniques," in *Proceedings of the IEEE Intelligent Transportation Systems Conference*, 2007, pp. 956 – 961.

Optimising High-dimensional Black-box Optimisation Problems in Soft Real-time Systems

Sebastian Niemann
Institute of Systems Engineering,
System and Computer Architecture,
Leibniz Universität Hannover,
Hanover, Germany
niemann@sra.uni-hannover.de

***Abstract*—The goal of this thesis is to provide solutions of high-dimensional black-box optimisation problems within a predefined deadline. While its virtual impossible in the case of most black-box problems to conclude the best solution within a deadline, instead a *good enough* solution is provided within a short amount of time. Additionally, a metric of certainty shall be developed to measure if the solution hold a substantial potential for improvement, given a bit more time to examine the underlying problem. However, this invokes two competing goals, since the number of function evaluations needs to be reduced in order to meet the deadline and also be large enough to meet the required quality of the solution as well as measure the certainty of the optimality conditions. Therefore, the aim is to harmonise these goals by making use of a priori knowledge about similar problems.**

I. Introduction

This thesis focuses on optimising high-dimensional black-box optimisation problems in soft real-time systems. In contrast to mathematical optimisation problems, black-box optimisation problems are missing a formal representation of their underlying objective function. These kind of problems often occur in real world systems, where the quality of a solution, which is measured by an objective function, can only be obtained by numerical approximation or statistical evaluations. Furthermore, a system is called a soft real-time system, if any non-compliance of the computation time of the system task and a predefined deadline results in a degrade of the systems quality.

Recognising a solution within a high-dimensional black-box optimisation problem to be within the best solutions is virtual impossible due to the large number of possible solutions and the absence of a formal representation of the objective function. In addition, deadlines within a narrow time frame compound this problem even further. Therefore, the contribution of this thesis did not concentrate on determining the actual best solution, but providing a *good enough* solution within a short amount of time. We consider thereby a solution to be good enough, if it was obtained within a predefined deadline and provides an acceptable quality.

An optimisation algorithm can hereby be described as a search algorithm that probes a sequence of solutions within a search space, with the aim to find a good enough solution. Commonly, the computation time of the optimisation task is assumed to depend only on the number of mutually distinct evaluations of the objective function. Therefore, an optimisation algorithm has to reduce the amount of functions evaluations, as the available computation time is shortened. By contrast, however, without any a priori knowledge regarding the optimisation problem, all clues that are available to direct the search towards an acceptable quality within black box optimisation problems are based on interpreting the quality of the sequence of probes only. For most high-dimensional optimisation problems, this invokes two competing goals: on the one hand, we want a small enough number of function evaluations to meet the deadline, while on the other hand we need a large amount of function evaluations to meet the required quality. Therefore, a metric of certainty of the optimality conditions shall measure if a solution hold a noteworthy room for improvement, given a bit more time to examine the underlying problem. This metric is then used to arbitrate between these conflicting goals.

The remainder of this paper elucidates how both goals can be harmonise by using a priori knowledge about similar optimisation problems. The paper is therefore organized as follows. Section II further describes the scope of this thesis and presents an application scenario that is currently focused. Section III presents and discusses the solution concept of this thesis. Afterwards, the related work is presented in section IV. This paper is concluded by a summary about the current process and what the next steps are going to look like in section V.

II. Scope and Application Scenario

This thesis is in the scope of self-optimising system, as seen in Organic Computing (OC) systems as well as Autonomic Computing (AC) systems. Evolutionary and more specific swarm based optimisation problems like the particle swarm optimisation and the genetic algorithm are analysed. It focuses on optimisation problems, which are induced by actual applications instead of synthetic benchmarking problems like the one induced by the Rosenbrock or Rastrigin function. Especially, applications are considered that allow to analyse a family of optimisation problems beforehand without any real-time restrictions, while the actual problem within the given family is only known during with soft real-time restrictions present.

The currently analysed application comes from the robotic sector. Figure 1 shows a so called 3(P)RRR parallel mech-

anism with kinematic redundancy. In comparison to classical serial mechanisms parallel kinematic machines (PKM) provide higher accuracy, higher stiffness, and better dynamic properties [1]. However, a main drawback of such mechanisms is the small workspace to installation space ratio. In order to exploit the maximal potential of the workspace, kinematic redundancy is used, realized by adding at least one actuated joint to one kinematic chain [2]. Therefore, the prismatic joint in Figure 1 allows to change position of the actuator in the lower right side. Informally, this possibility is meant by the term kinematic redundancy.

Fig. 1: Kinematically redundant 3(P)RRR mechanism

In order to achieve and to maximize the aforementioned potentials of kinematic redundancy, an appropriate optimization of the position of the additional actuator(s) is required for a predefined position of the manipulator in the middle of the mechanism. The solution needs to be obtained within one control cycle, which is about the length of one millisecond. Considering high-dimensional kinematic mechanisms with forty or more actuator, usually about half a million evaluations of the objective function are needed in order to resolve the kinematic redundancy, while each function evaluation needs to be calculated in two nano seconds in order to meet the deadline, disregarding any other computational cost. Missing out a deadline may reduce the potential of the mechanism drastically, while the system remains operational.

While the presented application scenario is quite specific, the results of this thesis can at least theoretically be applied to any application that satisfies the necessary conditions stated in the upcoming section. Which is actually the case for a lot of soft real-time systems that control any form of mechanical devices.

III. Solution Concept

The main goal of this thesis is to optimise high-dimensional black-box optimisation problems in soft real-time systems. As discussed in section I, this invokes two competing goals, which both need to be fulfilled. While this can not be done for increasingly smaller time frames without a priori knowledge, the solution concept is based on analysing similar optimisation problems beforehand, in order to use this knowledge to speed up the actual optimisation task during runtime.

Currently, two approaches are focused in order to conclude the similarity between optimisation problems: on the one hand, the similarity between optimisation problems is simply measured by sample solutions and there natural similarity defined by the metric of the value space of the objective function. On the other hand, the structure of both problems is compared with respect to there influence regarding a specific optimisation algorithm.

Optimisation tasks with similar properties are clustered, while a single representing optimisation problem of each cluster is further investigated in respect for all other problems. Therefore, the approach is limited to optimisation problems that are part of a large set of problems with a well-defined metric to conclude nearby optimisation problems within this set, in order to map an optimisation problem to a cluster. Concerning the discussed application in subsection II, the set of optimisation problems is induced by the set of all possible position of manipulator within the workspace. The metric is thereby defined by the euclidean distance between the positions that induce the problems.

With respect to the reduction of computation time needed to solve the optimization problem within a deadline, this goal can only be achieved by minimizing the complexity of the search space and/or of the objective function for the considered high-dimensional optimisation problems. The process is thereby organised as following:

1) Classification of similar problems
   - Sample-based
   - Structure-based
2) Reduction of the search space
   - Decomposition
   - Smoothing
   - ...
3) Enhancing optimisation algorithms
   - Runtime adaptation
   - Parallelisation
   - ...

While the sample-based classification works directly by the underlying metric of the value space of the objective function, the structure-based classification incorporates the influence of different objective functions on the performance of an optimisation algorithm.

After the classification of similar problems, the search space is reduced. Besides other methods that are concerned within the thesis, the correlation between different dimensions of the search space is studied and decomposed correspondingly in a set of optimisation problems with fewer dimensions. Another approach smooths the objective function in order to reduce the amount of local optima which make the optimisation task otherwise substantially more difficult.

Besides the parallelisation of optimisation algorithms, different optimisation algorithms are analysed in order to select the most suited algorithm during runtime. As stated by the no free lunch theorems for optimisation by Wolpert [3], there will never be a general purpose optimisation algorithm that performs better than a simple random search on average for all possible problems. Therefore, choosing an optimisation algorithm for black box optimisation problem without a priori knowledge is not guaranteed to achieve better results than a random search, independent of the given amount of time.

IV. Related Work

Concerning the presented application, several optimization algorithms to solve the resulting optimization problem have been applied [4], [5], [6], with particle swarm optimization (PSO) being the most suited algorithm [7], [8], [9]. To reduce the time-consuming aspect of the optimization task, Barbosa et al.[10] and Gao et al.[11] used an artificial neural network to approximate the calculation of the optimization criteria, achieving a remarkable reduction of the search time by 30% to 50%. However, none of these approaches, combined or used alone, is able to provide a satisfactory solution for the optimization problem within at least one hundred control cycles.

V. First Results and Next Steps

This paper presented a solution concept in order to optimise high-dimensional black-box optimisation problems in soft real-time systems. While this thesis is just at its beginning, first results for the presented application have already been submitted for publication. It was shown, that the reduction of the search space based on decomposition and sample-based classification reduced the computation time by one to two orders of magnitude. Furthermore, the reduced optimisation problem represented the actual problem as expected quite close, such that the solution of the reduced problem already fulfilled the required quality when measuring the solution within the actual optimisation problem. The next steps will deal with the classification of optimisation problem by comparing their structure in order to minimize the number of asimilar cluster of optimisation problems even further. Also the analysis of the behaviour of different optimisation algorithms started recently, in order to select to most suited one for a given problem during runtime and adapt the algorithm during different parts of the optimisation process.

References

[1] J.-P. Merlet, *Parallel Robots (Second Edition)*. Springer, 2006.

[2] M. G. Mohamed and C. M. Gosselin, "Design and analysis of kinematically redundant parallel manipulators with configurable platforms," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 277–287, June 2005.

[3] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, vol. 1, pp. 67–82, 1997.

[4] I. E. Moghaddam, "Kinematic redundancy in planar parallel manipulators," Ph.D. dissertation, University of New Brunswick, Canada, 2008.

[5] T.-S. Zhan and C.-C. Kao, "Modified pso method for robust control of 3rps parallel manipulators," *Mathematical Problems in Engineering*, vol. 1, p. 25, 2010.

[6] I. Tyapin, G. Hovland, and T. Brogardh, "Workspace optimisation of a reconfigurable parallel kinematic manipulator," in *Advanced intelligent mechatronics, 2007 IEEE/ASME international conference on*, sept. 2007, pp. 1 –6.

[7] A. Mishra and S. N. Omkar, "Singularity analysis and comparitive analysis of six degree of freedom stewart platform as a robotic arm by heuristic algorithms and simulated annealing." *International Journal of Engineering Science and Technology*, vol. 3, pp. 644–659, 2011.

[8] X.-S. Yang and S. Deb, "Cuckoo search via levy flights," in *World Congress on Nature Biologically Inspired Computing*, dec. 2009, pp. 210 –214.

[9] Q. Xu and Y. Li, "Stiffness optimization of a 3-dof parallel kinematic machine using particle swarm optimization," in *Robotics and Biomimetics, 2006. ROBIO '06. IEEE International Conference on*, dec. 2006, pp. 1169 –1174.

[10] M. Barbosa, E. Pires, and A. Lopes, "Optimization of parallel manipulators using evolutionary algorithms," in *Soft Computing Models in Industrial and Environmental Applications, 5th International Workshop (SOCO 2010)*, ser. Advances in Intelligent and Soft Computing. Springer Berlin / Heidelberg, 2010, vol. 73, pp. 79–86.

[11] Z. Gao, D. Zhang, and Y. Ge, "Design optimization of a spatial six degree-of-freedom parallel manipulator based on artificial intelligence approaches," *Robot. Comput.-Integr. Manuf.*, vol. 26, no. 2, pp. 180–189, Apr. 2010.

Calculating and Aggregating Direct Trust and Reputation in Organic Computing Systems

Rolf Kiefhaber
Institute of Computer Science
Augsburg University, Germany
E-Mail: kiefhaber@informatik.uni-augsburg.de

I. Introduction

The Organic Computing Initiative [1] identified the growing complexity of modern system as one of the big current challenges. These systems consist of a rising number of interacting parts, whose interactions increase in complexity as well. The Organic Computing Initiative aims to control these complexities by introducing so called self-x properties. The basic idea is to self-configure, self-optimize, self-heal and self-protect these systems. These properties are achieved by constantly observing the system and initiating autonomous reconfigurations when necessary (observer/controller paradigm [2]). By enabling autonomous reconfigurations Organic Computing Systems are able to react on disturbances without the immediate intervention of a user.

So far, Organic Computing Systems assume the benevolence of every involved interaction partner to obtain a more robust system utilizing these self-x properties. In open heterogeneous systems, like in cloud or grid computing, this benevolence assumption can no longer hold. In such systems, participants can enter and leave the systems at will. In addition, not every participant is interested in an altruistic cooperation to further the system goal. Some participants might try to exploit the systems or even try to attack and disrupt it.

By incorporating trust, the behavior of the participants can be monitored and identified. By utilizing this information the self-x properties of Organic Computing Systems are able to consider the behavior of its participants and are therefore able to maintain a more robust configuration in the face of unreliable components. This enables a reliable system out of unreliable components.

When speaking of trust, several definitions can be found in current literature. This dissertation is part of the research unit OC-Trust of the German Research Foundation (DFG). We published our definition of trust in [3]. We see trust as a multi-faceted multi-contextual subject and therefore defined the following facets:

- **Functional correctness:** The quality of a system to adhere to its functional specification under the condition that no unexpected disturbances occur in the system's environment.
- **Safety:** The quality of a system to be free of the possibility to enter a state or to create an output that may impose harm to its users, the system itself or parts of it, or to its environment.
- **Security:** The absence of possibilities to defect the system in ways that disclose private information, change or delete data without authorization, or to unlawfully assume the authority to act on behalf of others in the system.
- **Reliability:** The quality of a system to remain available even under disturbances or partial failures for a specified period of time, measured quantitatively by means of guaranteed availability, mean-time between failures, or stochastically defined performance guarantees.
- **Credibility:** The belief in the ability and willingness of a cooperation partner to participate in an interaction in a desirable manner. Also, the ability of a system to communicate with a user consistently and transparently.
- **Usability:** The quality of a system to provide an interface to the user that can be used efficiently, effectively and satisfactorily that in particular incorporates consideration of user control, transparency and privacy.

I focus on calculating reliability of nodes in a distributed network. When calculating trust, two categories have to be considered: Direct Trust and reputation.

- **Direct Trust** describes the trust one builds with an interaction partner based on its own experiences.
- **Reputation** stands for recommendations of third parties, i.e., the trust others had with my interaction partner.

In my thesis I investigate and research trust metrics to calculate direct trust, reputation and an aggregated total trust value from these two parts. The metrics are based on the trust definition mentioned above with focus on the facet reliability. The nodes form a heterogeneous open system. Each node is able to host some kind of service that provides functionality to use within the system. Integrating self-x properties in such a system enables a robust distribution of the services. Utilizing trust in this system, focused on the reliability of the nodes, enables a more robust distribution of the services, because unreliable nodes as well as node failures can now be considered when distributing services. It is thereby possible to rank the services by their importance and assign the more important services to more reliable nodes. Important services are those, which are essential for the functionality of the overlaying application. E.g., Bernhard el al. [4] present a computing grid to calculate big, yet parallelizable computational problems in a Multi-Agent System (MAS), which incorporates trust to form trusted communities (TCs). The managers, that administrate these TCs, are an example for an important service, since the

failure of a manager cripples the entire TC.

In my work I observe the behavior of nodes within a middleware. I assume that every node is equally able to implement the self-x properties. Therefore specific nodes for the self-x properties are not required, since each node implements the trust metrics and the algorithms for the self-x properties. In addition, I investigate systems, where the reliability of the nodes can be different. If all nodes would be reliable, nothing bad could actually happen or would be quite unlikely, e.g, a node failure, therefore no trust would be needed. The middleware system investigated in this work, the so called *Trust Enabling Middleware (TEM)* [5], is supposed to handle unreliable components and can be applied to any kind of distributed system, i.e., Multi Agent Systems (MAS). The TEM implements the algorithms developed in this dissertation and provides interfaces to allow all applications running on the TEM to use these algorithms.

II. Metrics

To calculate the trust values required for the self-x properties, four different parts have to be considered:

1) **Direct Trust**: First, the reliability of the nodes has to be observed and calculated. This is the basis for the other trust metrics and the decisions of the self-x properties.
2) **Reputation:** If the personal experiences with other nodes are not adequate enough to form a consistent decision, the experiences of other nodes have to be obtained. Therefore a reputation mechanism has to be defined.
3) **Confidence:** Before both values, direct trust and reputation, can be aggregated to a total trust value, the reliability of one's own trust value has to be determined, the so called *confidence*. If a node does have a direct trust value but is not confident about its accuracy, it needs to include reputation data as well.
4) **Aggregation:** When all the aforementioned values are obtained, a total trust value based on the direct trust and reputation values can be calculated using confidence to weight both parts against each other. This value can then be used to improve the self-x properties.

While direct trust developed in this dissertation is focused on obtaining the reliability of nodes, the reputation, confidence and aggregation are applicable to all kinds of direct trust values of any facet. The metrics are generic enough to achieve this goal.

A. Direct Trust

The basis for the trust value is the direct trust, the trust based on the direct experiences of a node. For an improvement of the self-x properties an evaluation of the reliability of a node is required. Such an estimation has to be done without knowledge about the functionality of the distributed services, since this estimation is done on middleware level. Nevertheless the reliability of a node can be measured by observing the message flow to other nodes. If messages are lost, either the node or the connection to it is unstable or has failed. In this case the reliability of the node is rated down and it is no longer appropriate for important services, because messages targeted to such a service might be lost as well. Additionally the loss of messages might refer to an error in the node itself and its imminent failure. In this case an important service running on it would fail as well.

B. Reputation

If no direct experiences could be obtained, and therefore no direct trust value calculated, or if the direct trust value is not yet sufficient enough, other nodes that already had experiences with a node are asked about their opinion. The total amount of all opinions of other nodes forms the reputation value for the potential interaction partner. A node n_1 that has direct experiences with another node n_2 is called a *neighbor* of n_2. An important aspect of the reputation metric is to separate the direct trust value of a node in any context from its ability to provide appropriate reputation data. Marmól and Pérez [6] demonstrated attack scenarios, which are only possible, if these values are not separated. In general, the reliability of a node says nothing about the accuracy of its recommendations. The recommendation of a neighbor can be weighted for the total reputation value (consisting of the recommendations of all neighbors about a node) regarding its previous recommendations. When the information of a neighbor has proven to be false, e.g., the node tries to disrupt the system by providing incorrect information, its future recommendations are rated down for the total reputation value. Using this method, lying nodes can be identified and their incorrect recommendations discarded. This also means, that nodes can redeem themselves by providing correct information in the future.

A neighbor which recommendations differ from a node's own experiences is not necessarily malicious. Its experiences might be different from the nodes, e.g., the connection between it and its neighbor is unstable. Its recommendation is nonetheless of no use for the node, since the direct experiences of the node take precedence to decide whether to actually interact with the interaction partner. Golbeck [7] demonstrated in her evaluation scenario, which consisted of a movie rating platform combined with trust relations between the raters, that getting recommendations from others, that have similar experiences, or in this case taste in movies, is superior than using the opinion of the masses.

The thresholds, when an recommendation is similar enough to one's own experience, are adjustable as well as the amount of maximal adjustment of the weight, be it positive or negative, is adjustable. Therefore the metric can be adjusted to any kind of application scenario.

C. Confidence

An estimation about the accuracy of one's own trust value is required, before both values can be aggregated. This estimation is done by calculating the *confidence* of the direct trust value. With a high confidence, the direct trust will be rated higher than reputation in the total trust value and vice versa. The confidence rates three different aspects of the experiences that were used to calculate the trust value:

- **Number**: Very few experiences are not suitable to express the actual behavior of an interaction partner, especially when its behavior contains some variance.
- **Age**: Older experiences might be outdated when interaction partner are able to change their behavior. Such outdated data might reflect its past behavior but not the current one.
- **Variance**: Since trust values are typically calculated by a mean or weighted mean metric a high variance of the experiences is not reflected in such a mean value. Therefore it is important to consider the variance as well.

The total confidence is a weighted mean of these three aspects, whereas the weights are able to be adjusted by the application to adhere to scenarios, where one or more of these aspects are less important than the others. Additionally the number and weight confidence are adjustable as well. For the number confidence, the threshold, when enough experiences are gathered, is adjustable. For the age confidence, the thresholds, when an experience is completely outdated or completely up to date is adjustable as well.

D. Aggregation

With the confidence a weight for the aggregation using a weighted mean of direct trust and reputation can be calculated. A high confidence results in a high weight for the direct trust value and vice versa. Here the question is to find a good formula to calculate the weight from the confidence. Figure 1 illustrates the function I want to use to archive this goal. When looking at human trust decisions most of it is done intuitively without any form of quantification. In my thesis I plan to quantify the point, when to switch from reputation to direct trust, or more precisely, how to calculate the weight between direct trust and reputation. In my formula this means to find good values for the two thresholds τ_{cl} and τ_{ch}.

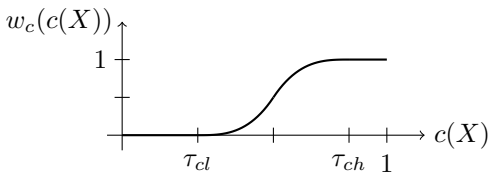

Fig. 1. Function to calculate the weight $w_c(c(X))$ for the total trust value based on the confidence $c(X)$ of all experiences X

III. Related Work

Trust is an actively worked on research fields with a plethora of different metrics. Many of them include users for direct trust. Below are some of the more prominent trust frameworks presented.

SPORAS [8] is another reputation metric. Its focus is to prevent entities to leave and rejoin the network to reset possible bad reputation values. Compared to my reputation metric, SPORAS does not assign different values for the reputation value provided by another interaction partner and the trustworthiness of that interaction partner to give accurate reputation data. The trustworthiness is calculated from its reputation value. I differentiate between these values by defining separate weights; Mármol and Pérez [6] have shown the importance to do this.

FIRE [9] is a trust framework combining direct trust and reputation (called *witness reputation* in FIRE). In addition, it adds the trust parts of *certified trust* and *role-based trust*. Certified trust describes past experiences others had with an agent, who can present it as reference of his past interactions. Role-based trust stands for generic behavior of agents within a role and the underlying rules are handcrafted by users. The four parts are then aggregated with a weighted mean, whereas the weights are adjusted by a user depending on the current system. In comparison, my work does not require user handcrafted parts like the role-based trust of FIRE and is therefore able to run in a fully automated environment.

ReGreT [10][11] is a trust framework providing similar metrics for direct trust, reputation, and aggregation to my metrics. Some differences to my work exist. The age of experiences is part of the direct trust calculation whereas I have the age, number and variance as confidence (called the *reliability of the trust value* in ReGreT). Additionally, my metrics for the confidence metrics are parametrized. Similarly, my reputation metric can be parametrized to define the threshold, when one's own experiences are close enough to the reputation data given by a neighbor (called a *witness* in ReGreT). Also I do not use the confidence directly for the aggregation but a parameterizable function to calculate the weight for using direct trust instead of reputation, beside using a non linear function to aggregate direct trust and reputation. One of the major differences though lies in the evaluation. While ReGreT works in a scenario with fixed agent behaviors I investigate systems with varying behavior, where a very trustworthy node can change to the direct opposite. Several such changes per scenario are considered by me.

I also investigate the impact of the parameters and identify appropriate parameter configurations by utilizing automatic design space exploration.

IV. Evaluation

To evaluate the different metrics, especially the aggregation, a scenario with a set of nodes is defined, where the nodes each have a mean reliability with a specific variance. These two values, mean reliability r and variance v, are generated randomly for a scenario, but within certain bounds, e.g., more reliable nodes have $r \in [0.8, 0.9]$ and highly unreliable nodes have $r \in [0.2, 0.3]$. This behavior is achieved by utilizing a beta distribution[1]. The result of each interaction, and therefore the rating of the experience, is taken from the beta distribution. Jøsang and Elouedi [12] presented subjective logic, which enriches binary logic with uncertainty and adds a complete algebra on it. They showed that subjective logic expressions can be bidirectionally translated to a beta distribution. Since trust is used to handle uncertainties, using a beta distribution for node behaviors is suitable. Additionally a beta distribution includes several other distributions, e.g., $\alpha = 1, \beta = 1$

[1] https://en.wikipedia.org/wiki/Beta_distribution

for mean distribution, so several possible behaviors can be modeled.

To investigate the effects of the different metrics, there is an amount of nodes to interact with n and some other nodes (the evaluation nodes) that interact with these nodes n_e. The simulation is divided into time steps, where each evaluation node is performing an interaction with one of the normal nodes. They consider their own experiences, as well as the information from the other evaluation nodes, to decide which node to interact with. Some nodes also change their behavior, i.e., changing the configuration of their beta distribution, several times in the evaluation. The evaluation nodes thereby try to obtain the best result from each interaction, which is called *benefit*. The benefit represents the rating of a single experience. It can be a number between 0 (worst possible result of the interaction) to 1 (best possible result of the interaction. After several time steps the total cumulative benefit is taken as fitness function to rate the effectiveness of each metric variant, since the goal of a node is to choose the participants that provide the best benefit and therefore highly profitable interactions. These variants include using only direct trust (DT), using direct trust and confidence (DTC) as well as combining all parts, that is direct trust, confidence and reputation (DTCR).

Since nearly all parts of the metrics can be parametrized, except direct trust, an *automated design space exploration (ADSE)* is applied to find suitable parameters for each metric as well as investigating the effects of poorly chosen parameters. An ADSE employs heuristic algorithms, like genetic or particle swarm optimization algorithms, to find good enough results in a parameter space that is too big to traverse completely. Thereby several scenarios of agent behavior should be investigated, including more frequent behavior changes or completely random behavior. An important aspect is also the selection metric used. To balance the exploration versus exploitation problem (when should unknown interaction partners be explored versus when to use already known interaction partners) a selection metric based on the roulette-wheel selection metric is applied.

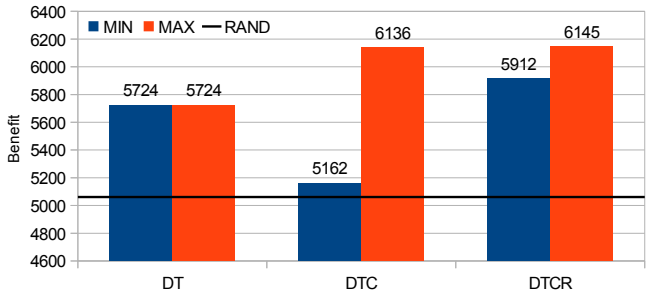

Fig. 2. Results when comparing the effectiveness of the different trust metrics with $n = 100$ and $n_e = 10$

Figure 2 shows some results of an evaluation when using only direct trust (DT), direct trust and confidence (DTC) as well as direct trust, confidence and reputation (DTCR) compared to random (RAND) to choose the next interaction partner. As was described before the goal was to maximize the total cumulative benefit over all interactions, in case of this simulation 8000. The metrics were parametrized using ADSE to find the worst possible solution (left column / MIN) and best possible solution (right column / MAX) for these parameters. It can be seen that using trust is better than random in all cases. Adding confidence to direct trust can increase the total benefit significantly but the benefit can get worse than by using direct trust alone, if unfitting parameters are defined. Adding reputation balances bad parameter choices while maintaining a high maximum result.

ACKNOWLEDGMENT

This research is sponsored by the research unit OCTrust (FOR 1085) of the German Research Foundation (DFG).

REFERENCES

[1] C. Müller-Schloer, "Organic computing: on the feasibility of controlled emergence," in *Proceedings of the 2nd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS 2004, Stockholm, Sweden, September 8-10, 2004*, A. Orailoglu, P. H. Chou, P. Eles, and A. Jantsch, Eds. ACM, 2004, pp. 2–5.

[2] U. Richter, M. Mnif, J. Branke, C. Müller-Schloer, and H. Schmeck, "Towards a generic observer/controller architecture for Organic Computing," in *Informatik 2006 - Informatik für Menschen, Band 1, Beiträge der 36. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 2.-6. Oktober 2006 in Dresden*, ser. LNI, vol. 93. GI, 2006, pp. 112–119.

[3] J.-P. Steghöfer, R. Kiefhaber, K. Leichtenstern, Y. Bernard, L. Klejnowski, W. Reif, T. Ungerer, E. André, J. Hähner, and C. Müller-Schloer, "Trustworthy organic computing systems: Challenges and perspectives," in *Autonomic and Trusted Computing*, ser. Lecture Notes in Computer Science, B. Xie, J. Branke, S. Sadjadi, D. Zhang, and X. Zhou, Eds. Springer Berlin / Heidelberg, 2010, vol. 6407, pp. 62–76.

[4] Y. Bernard, L. Klejnowski, J. Hähner, and C. Müller-Schloer, "Towards Trust in Desktop Grid Systems," in *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), 2010*, 2010, pp. 637–642.

[5] G. Anders, F. Siefert, N. Msadek, R. Kiefhaber, O. Kosak, W. Reif, and T. Ungerer, "TEMAS – A Trust-Enabling Multi-Agent System for Open Environments," Universität Augsburg, Tech. Rep. 2013-04, April 2013. [Online]. Available: http://opus.bibliothek.uni-augsburg.de/opus4/frontdoor/index/index/docId/2311

[6] F. G. Mármol and G. M. Pérez, "Security threats scenarios in trust and reputation models for distributed systems," *Computers & Security*, vol. 28, no. 7, pp. 545–556, 2009.

[7] J. A. Golbeck, "Computing and applying trust in web-based social networks," Ph.D. dissertation, University of Maryland, College Park, MD, USA, 2005.

[8] G. Zacharia, "Trust management through reputation mechanisms," *Applied Artificial Intelligence*, vol. 14, pp. 881–907, 2000.

[9] T. Huynh, N. R. Jennings, and N. Shadbolt, "An integrated trust and reputation model for open multi-agent systems," *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 13, no. 2, pp. 119–154, 2006.

[10] J. Sabater and C. Sierra, "Social ReGreT, a reputation model based on social relations," *SIGecom Exch.*, vol. 3, no. 1, pp. 44–56, Dec. 2001. [Online]. Available: http://doi.acm.org/10.1145/844331.844337

[11] J. Sabater Mir, "Trust and reputation for agent societies," Ph.D. dissertation, Universitat Autònoma de Barcelona, 2002.

[12] A. Jøsang and Z. Elouedi, "Redefining material implication with subjective logic," in *Proceedings of the 14th International Conference on Information Fusion (FUSION 2011)*, 2011, pp. 1–6.

Improving the Self-X Properties of Organic Computing Systems with Trust

Nizar Msadek
Department of Computer Science
University of Augsburg
D-86159 Augsburg, Germany
Email: Nizar.Msadek@informatik.uni-augsburg.de

I. Introduction

The Organic Computing Initiative [1] has turned out to be a challenging vision for future information processing systems. This initiative consists of developing computer systems capable of so-called self-x properties (like self-configuration, self-optimization, self-healing and self-protection) to cope with the rapidly growing complexity of computing systems and to reduce the barriers that complexity poses to further growth. These properties are achieved by constantly observing themselves and initiating autonomous reconfiguration if necessary.

An essential aspect that becomes particulary prominent in this kind of systems is trust [2]. As part of my PhD thesis, a new design of self-x properties for organic computing systems will be investigated. Its main task is to improve self-x properties with trust capabilities to enable building a reliable system from unreliable components. The middleware system used in this work is the Trust-Enabling Middleware (TEM [3]) but these techniques can also be applied to any kind of distributed system.

This dissertation is part of the research unit OC-Trust of the German Research Foundation(DFG), which presented the following trust metrics to calculate the trust values required for the self-x properties. It is to note that all these trust metrics are integrated in TEM.

- **Direct Trust [4]** Is based on the experiences one has made directly with an interaction partner. Typically, trust values are calculated by taking the mean or weighted mean of past experiences.
- **Reputation [5]** Is based on the trust values of others that had experiences with the interaction partner. Reputation is typically raised if not enough or outdated experiences exist.
- **Confidence [6]** Before both values, direct trust and reputation, can be aggregated to a total trust value, the reliability of one's own trust value has to be determined, the so called confidence. If a node does have a direct trust value but is not confident about its accuracy, it needs to include reputation data as well.
- **Aggregation [7]** When all the aforementioned values are obtained, a total trust value based on the direct trust and reputation values can be calculated using confidence to weight both parts against each other. This value can then be used to improve the self-x properties.

The remainder of this paper is structured as follows. Section II introduces self-configuration, Section III gives an overview of self-optimization, Section IV describes mechanisms of self-healing and how to cope with failures. Section V presents the role of self-protection. Conflicting problems of trust values are discussed in Section VI. Finally, the scalability is shown in Section VII.

II. Self-Configuration

The approach of self-configuration is a crucial part for developing dependable and robust systems using self-x properties. This consists mainly of finding a robust distribution of services by including trust. The services are therefore categorized into important services i.e., with high trust level, and non important services i.e., with low trust level. The goal is to maximize the availability of important services. Therefore, it is necessary to assign the more important services to more reliable nodes. In addition to the reliability, resource requirements (e.g., like CPU and memory) should also be considered to be able to balance load of the nodes.

A. Metrics

The self-configuration focuses on assigning services with different trust levels to nodes such that the more important services are assigned to the more reliable nodes. Furthermore, the overall utilization of resources in the network should be well balanced. Therefore a metric is defined to calculate a Quality of Service (QoS_{total}).

$$QoS_{total} = (1 - \alpha) \cdot QoS_{trust} + \alpha \cdot QoS_{workload}.$$

The relationship between trust and workload can be set through α. α is constant $(0 \leq \alpha \leq 1)$ for a node. If $\alpha = 1$, the QoS_{total} is only obtained by the current value $QoS_{workload}$. If $\alpha = 0$, the QoS_{total} is decided only by the actual QoS_{trust} value. A higher value α favors $QoS_{workload}$ over QoS_{trust}.

- $\mathbf{QoS_{trust}}$ indicates how well the reliability of a node fulfilled the required reliability of a service. Figure 1 visualizes the possible situations that can occur in the calculation of the QoS_{trust}.

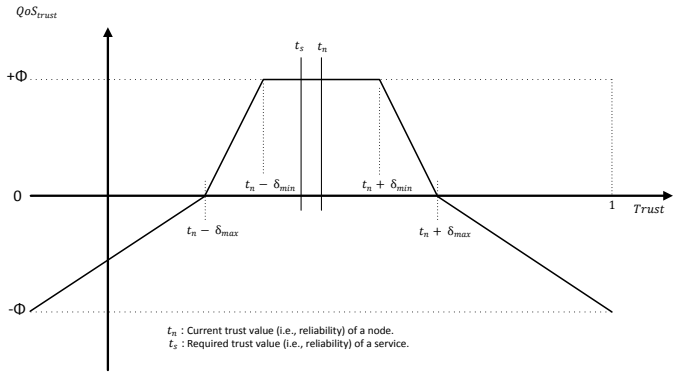

Fig. 1. Calculating QoS_{trust} based on the difference between the reliability of the node n (t_n) compared to the required reliability of the service s (t_s)

t_n represents the current trust value of a node n. t_s represents the required trust value of service s. If both of these values are close enough then n has fulfilled the required trust value of a service s. Close enough is defined by the threshold δ_{min}. If the difference between t_n and t_s is more than δ_{min}, then QoS_{trust} will be gradually decreased until it reaches $t_n \pm \delta_{max}$. If t_s is even beyond $t_n \pm \delta_{max}$ then the QoS_{total} will be fully decreased by ϕ where ϕ is the maximum value QoS_{trust} can decrease.

- $\mathbf{QoS_{workload}}$ is computed with the metric of Trumler [8]. As long as a node is not overburdened, the quality of service decreases linearly, otherwise it decreases exponentially.

B. Self-Configuration Process

This section discusses the methodology for distributing services. This consists of a collection of services with different priority levels which should run on nodes with different reliability levels. It is known to be a NP-hard problem to find an optimal solution for the distribution of the services on the nodes, such that the quality of service is optimal in terms of some predefined parameters [9]. Furthermore, there is no known polynomial algorithm which can, for a given solution, identify whether it is optimal. The aim behind self-configuration is to find a distributed and robust but not necessarily optimal, solution.

The quality of service metric presented in II-A is intended to evaluate the distribution phase which is based on the Contract Net Protocol [10]. During the distribution phase, every node on the network can act at different times or for different services as a manager or contractor. A *manager* is responsible for assigning services. A *contractor* is responsible for actual execution of the service. However, the manager is determined earlier by the user. Figure 2 visualizes a step-by-step example on how the negotiation process is run between nodes.

1) **Service Announcement:** The manager (e.g., node1) that has a service initiates contract negotiation by advertising the existence of that service to the other contractors (e.g., node2, node3 and node4) with a service announcement message. A service announcement can be transmitted to a single contractor in the network (unicast), to a specific set of contractors (multicast) or to all contractors (broadcast).
2) **Bidding:** Every contractor that receives the announcement calculates the $QoS_{workload}$ itself for the given service, i.e., based on its own locally available resources and then submit its bid in form of $QoS_{workload}$ to the manager back. Note that the service annoucement is ignored if the service cannot be hosted due to missing resources.
3) **Awarding:** If the expiration time has passed, the manager that sends the service announcement must calculate QoS_{trust} for every contractor in order to build the QoS_{total} and decides who to award the contract to. The result of this process will be then communicated to the contractors that submitted a bid. The expiration time is defined as a deadline for receiving bids. It is to note that the expiration time is determined earlier by the user.

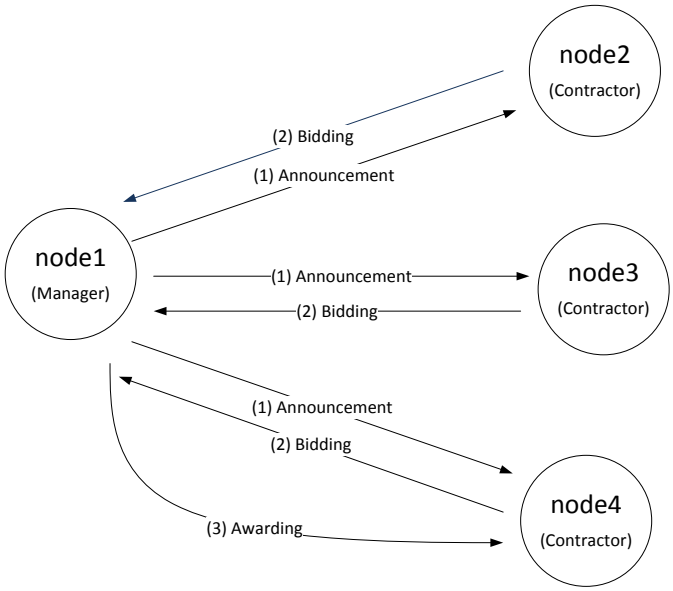

Fig. 2. Elementary representation of the distribution phase

C. Conflict Resolution

During the self-configuration process, several nodes could be ranked with the same QoS_{total}. This might lead to a conflict for the manager to decide to whom he awards the service. To avoid this a conflict resolution mechanism is used which does not need any further messages. The conflict resolution mechanism consists of three stages which might be used in the following chronological order:

1) **Minimum latency:** The node with the lowest latency will get the service.
2) **Minimum amount of already assigned services:** The node with the least amount of already assigned services will get the service, assuming that a lower amount of

services will produce less load (e.g., process or thread switching produces additional load).

3) **Node ID:** It is unlikely but not impossible that all of the former values were equal. In this case the id of the node will be used to find a solution to the conflict because every node has a unique id.

III. Self-Optimization

Based on the proposed self-configuration techniques, the services can be distributed on the nodes by different distribution strategies:

- **Uniform distribution:** The services are distributed on the nodes to evenly utilize all nodes and prevent single nodes to be overburdened. This leaves every node with a safety margin to cover possible performance spikes.
- **Power save distribution:** All services should run on a minimal number of nodes, so that free nodes can be deactivated in order to save energy.

Without trust, important services might run on unreliable nodes and are prone to failures. Such situations can be avoided. With trust, the reliability of a node can be measured and taken into consideration for the service distribution. For that reason, the distribution mechanisms should be investigated with and without a known reliability. The differences between using trust and not using trust have to be evaluated regarding the downtime of important services.

IV. Self-Healing

To investigate and research Self-Healing metrics, two ways have to be considered:

- **Proactive Self-Healing:** Enables to detect node instability prior to fail and then to move all running services by using self-configuration techniques to a more reliable node. False proactive shifts should be avoided.
- **Reactive Self-Healing:** Nodes save recovery information periodically during failure free execution. Upon failure, which has to be detected by using a failure detector, a node uses the already saved information to restart from an intermediate state i.e., called checkpoint, thus reducing the amount of lost computation.

V. Self-Protection

Trust values build the basis for all operations to increase the robustness of an organic computing system. Therefore, they must be specially protected against manipulation. Mármol and Pérez [11] presented some of the most important and critical security threat scenarios that can be found in the area of trust and reputation in a distributed system. Hence, all these scenarios have to be investigated and researched in order to make the self-x properties more resistent against such attacks.

VI. Problem of conflicting trust values

Another interesting point is to find a solution for conflicting trust values. This can happen by collecting reliability values independently from the neighbors of a node that can contradict each other. Figure 3 visualizes this problem in an example of three nodes.

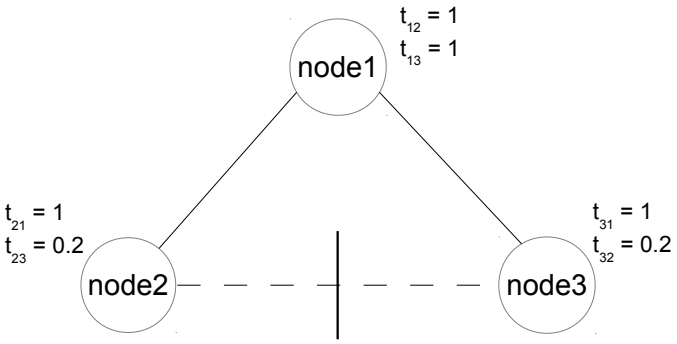

Fig. 3. Trust conflicting values in an example of three nodes

A shielding wall is set between two nodes i.e., node2 and node3 producing poor reliability values between these nodes, while a third node (node1) is not affected. t_{23} is the trust value node2 has about node3, so it wants to apply self-healing techniques in order to save all services running on node3, while node1 sees no need for action. Such situations must be omitted using metrics, which enable to deal with conflicting values.

VII. Dealing with large scale organic computing systems

In a hierarchical system, e.g., in a clustered Data Center, the already developed trust metrics are so far not entirely applicable, since only the next level in the hierachy is visible. Some nodes within a cluster could be less trustworthy. The cluster itself is still trusted because the cluster head is able to deal with its cluster members. Such situations can be omitted by using methods enabling the cluster head to control unreliable cluster members and to return a good result despite such members.

VIII. Related Work

The presented trust-enhanced self-x properties differ from state of the art selforganising mechanisms [12] [13] [1] [14] in three major points:

1) Development of techniques that allow the consideration of trust during analysis and interaction of Organic Computing systems.
2) Possibility to control complex systems with variable behavior.
3) Making a reliable and robust system out of unreliable components.

IX. Summary and Outlook

In this paper, a new design of self-x properties for organic computing systems is presented. Its main task is to improve using trust self-configuration, self-optimization, self-healing and self-protection. This approch will be embedded in a future work into the TEM, which is a trust enabling middleware implemented in Java and based on a peer to peer network.

Furthermore, two major existing problems in TEM are discussed, which are scalability and conflicting trust values.

ACKNOWLEDGMENT

This research is sponsored by the research unit OCTrust (FOR 1085) of the German Research Foundation (DFG).

REFERENCES

[1] C. Müller-Schloer, "Organic Computing - On the Feasibility of Controlled Emergence," *CODES + ISSS 2004. International Conference on Hardware/Software Codesign and System Synthesis, 2004.*, vol. 2-5, 2004.

[2] J.-P. Steghöfer, R. Kiefhaber, K. Leichtenstern, Y. Bernard, L. Klejnowski, W. Reif, T. Ungerer, E. André, J. Hähner, and C. Müller-Schloer, "Trustworthy Organic Computing Systems: Challenges and Perspectives ," *Proceedings of the 7th International Conference on Autonomic and Trusted Computing (ATC 2010), Springer*, vol. 14, pp. 62–76, 2010.

[3] G. Anders, F. Siefert, N. Msadek, R. Kiefhaber, O. Kosak, W. Reif, and T. Ungerer, "TEMAS - A Trust-Enabling Multi-Agent System for Open Environments," Universität Augsburg, Tech. Rep., 2013.

[4] R. Kiefhaber, B. Satzger, J. Schmitt, M. Roth, and T. Ungerer, "Trust measurement methods in organic computing systems by direct observation," in *IEEE/IFIP 8th International Conference on Embedded and Ubiquitous Computing (EUC 2010)*, december 2010, pp. 105 –111.

[5] R. Kiefhaber, S. Hammer, B. Savs, J. Schmitt, M. Roth, F. Kluge, E. André, and T. Ungerer, "The neighbor-trust metric to measure reputation in organic computing systems," in *Fifth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW 2011)*, october 2011, pp. 41 – 46.

[6] K. Rolf, A. Gerrit, S. Florian, U. Theo, and R. Wolfgang, "Confidence as a means to assess the accuracy of trust values," in *The sixth IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO2012)*, september 2012.

[7] K. Rolf, J. Ralf, M. Nizar, and U. Theo, "Ranking of direct trust, confidence, and reputation in an abstract system with unreliable components," in *The seventh IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO2013)*, september 2013.

[8] W. Trumler, "Organic ubiquitous middleware," Ph.D. dissertation, 2006.

[9] K. R. Reischuk, *Komplexitätstheorie: Band 1*. Teubner Verlag, 1999.

[10] R. G. Smith, "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver ," in *Defence Research Establishment Atlantic*. IEEE TRANSACTIONS ON COMPUTERS,, 1980, pp. 1–10.

[11] F. G. Mármol and G. M. Pérez, "Security threats scenarios in trust and reputation models for distributed systems," *Computers & Security*, vol. 28, no. 7, pp. 545–556, 2009.

[12] R. Urban, M. Moez, B. Jürgen, M.-S. Christian, and S. Hartmut, "Towards a generic observer/controller architecture for organic computing," *Bonner Köllen Verlag*, pp. 112–119, 2006.

[13] O. Jeffrey and C. David M, "The vision of autonomic computing," *IEEE Computer Society*, pp. 41 – 50, 2003.

[14] B. Jürgen, M. Moez, M.-S. Christian, P. Holger, R. Urban, R. Fabian, and S. Hartmut, "Organic computing addressing complexity by controlled self-organization," *Second International Symposium on Leveraging Applications of Formal Methods, Verification and Validation*, pp. 185 – 191, 2006.

Call for participation in the Doctoral Dissertation Colloquium (DDC)

organised by the Special Interest Group on Organic Computing within the Gesellschaft für Informatik in Augsburg, Germany (May 23 – May 24, 2013).

http://www.informatik.uni-augsburg.de/lehrstuehle/oc/OC-DDC13

We invite applications from PhD students at any stage of their doctoral studies on Organic Computing, Self-Organisation and related research topics.

The intent of the DDC is to bring together PhD students who have their research focus within the broader Organic Computing community (mainly at the beginning of their PhD period) and will not have defended their thesis before May 2013. The main goal of this colloquium is to foster excellence in OC-related research by providing students feedback and advice that are particularly relevant to their doctoral studies and career development. Each participant will take part in the colloquium organisation and the review process for the other participants. The schedule includes:

- Prepare an extended abstract of 3 pages reflecting your PhD concept as follows: /informatik/de/lehrstuehle/oc/downloads/IEEEtran.zip
- Participate in the Review Process
- Participate in the (supervised) decision process (acceptance of submissions)
- Present your work to an audience consisting mainly of other PhD students from your community
- Identify points of contact between your work and others
- Discuss your research in a less formal atmosphere compared to a conference
- Build your social research network (e.g. at the evening event)
- Listen to invited talks of experts in your research domain
- Enjoy the opportunity to get hints on the PhD process and related topics from these experts

PhD. students are invited to participate by submitting an extended abstract of up to *3 pages* (written in English, in the posted IEEE format). This colloquium will give participants the opportunity to present their ongoing research in a friendly forum. They will obtain valuable feedback from colloquium attendees in a constructively critical and informal atmosphere.

In addition, two invited talks are part of the colloquium:

- **Prof. Dr.-Ing. Martin Hoffmann**, Fachhochschule Bielefeld and Volavis GmbH: received his PhD in 2009 from Leibniz Universität Hannover. Prof. Hoffmann will talk about Smart Camera Networks as one promising application area for Organic Computing systems as well as his experiences as a PhD student.
- **Prof. Dr. Bernhard Sick**, Universität Kassel: leads the intelligent embedded systems group at Universität Kassel. Prof. Sick will talk about his current OC-related research topics as well as the PhD process from the perspective as the dissertation adviser.

Important Dates:

Doctoral Dissertation Extended Abstract Submission Deadline: **April 23, 2013 (extended deadline)**

Review Deadline: May 03, 2013
Registration Deadline: May 10, 2013
Colloquium Dates: May 23 – 24, 2013

Submission:

Submission of 3 pages extended abstract via EasyChair using the following link:
https://www.easychair.org/conferences/?conf=ocddc13

Colloquium Details:

- Participation cost (includes lunch, drinks, etc.): 55 EUR (you will receive a bill in advance)
- Participants have to organise travel and accomodation themselves (Hotel contingents will be available)
- Please contact the following hotels by mentioning "Uni Augsburg" in the reservation.
- Hotel Ibis, room charge: 69,- € including breakfast
  http://ibishotel.ibis.com/de/hotel-1438-ibis-augsburg-hauptbahnhof/index.shtml
- B&B Hotel, room charge: 52,- € excluding breakfast
  http://www.hotelbb.de/de/augsburg

Doctoral Dissertation Co-chairs:

- Prof. Dr. Jörg Hähner (Universität Augsburg, Germany)
- Prof. Dr.-Ing. Martin Hoffmann (Fachhochschule Bielefeld)
- Prof. Dr.-Ing. Christian Müller-Schloer (Leibniz Universität Hannover, Germany)
- Prof. Dr. Bernhard Sick (Universität Kassel, Germany)
- Dr.-Ing. Sven Tomforde (Universität Augsburg, Germany)
- Dr. Hella Seebach (Universität Augsburg, Germany)

Doctoral Dissertation Colloquium, Universität Augsburg

Thursday, 23. Mai 2013

09:00 Uhr	09:15 Uhr	Welcome	Sven Tomforde	
09:15 Uhr	10:00 Uhr	Keynote	Martin Hoffmann	
10:00 Uhr	11:00 Uhr	Session 1	Ioannis Zgeras	CASEP - Code Analysis, Speedup Estimation and Parallelization
			Oliver Mattes	An Autonomous Self-Optimizing Memory System for Upcoming Manycore Architectures
11:00 Uhr	11:15 Uhr	Pause		
11:15 Uhr	12:45 Uhr	Session 2	Stefan Rudolph	A Distributed Controller for Organic Computing Applications
			Benedikt Eberhardinger	Model-based, Adaptive Testing of Organic Computing Systems
			Christian Krupitzer	FESAS: A Framework for Engineering Self-Adaptive Systems
12:45 Uhr	13:30 Uhr	Lunch Break		
13:30 Uhr	15:00 Uhr	Session 3	Martin Jänicke	Self-Adaptation of Multi-Sensor-Systems with Organic Computing Techniques
			Katharina Stahl	AIS-based Anomaly Detection in Self-x Systems
			Tobias Reitmaier	Active Learning of Generative and Discriminative Classifiers for Organic Computing
15:00 Uhr	15:15 Uhr	Pause		
15:15 Uhr	16:45 Uhr	Session 4	Michael Roth	Distributed Management of Cloud Computing Applications
			Matthias Sommer	Towards a decentralized intelligent traffic management system
			Sebastian Niemann	Optimising High-dimensional Black-box Optimisation Problems in Soft Real-time Systems
16:45 Uhr	18:00 Uhr	Working Groups		
19:00 Uhr		Social Event		

Universität Augsburg University

ORGANIC COMPUTING

Doctoral Dissertation Colloquium, Universität Augsburg

Friday, 24. Mai 2013

09:00 Uhr	09:45 Uhr	Keynote	Bernhard Sick	
09:45 Uhr	10:45 Uhr	Session 4	Nizar Msadek	Improving the Self-X Properties of Organic Computing Systems with Trust
			Rolf Kiefhaber	Calculating and Aggregating Direct Trust and Reputation in Organic Computing Systems
10:45 Uhr	11:00 Uhr	Pause		
11:00 Uhr	12:00 Uhr	Discussion of Working Groups		
12:00 Uhr	12:30 Uhr	Summary		